Singularite ou l'émergence d'un paradoxe fécond

Francois Bourdon

SINGULARITÉ

OU L'ÉMERGENCE D'UN PARADOXE FÉCOND

TESTAMENT DE RECHERCHE

2019 - 2023

François Bourdon

Chercheur associé au GREYC

Université Caen Normandie

Normandie Université — UNICAEN Université Caen Normandie — ENSI CAEN — cnrs — GREYC Laboratoire de recherche en sciences du numérique

Marcel Proust dans une lettre à son ami Daniel Halévy:
«C'est à la cime du particulier qu'éclôt l'universel.»[1]

1 Rapporté p. 189 dans le livre « L'autre qu'on adorait » de Catherine Cusset aux éditions Folio.

«Je ne peux pas voir mes sculptures ou mes peintures de la même façon que les spectateurs. Pour ces derniers, si elles sont exposées, c'est qu'elles sont finies, alors que pour moi, elles ne sont jamais finies.»

«Je cherche à commencer la sculpture, car c'est quelque chose qui m'était étrangère, que je ne comprenais pas. C'est la raison pour laquelle je suis pressé de réessayer sans cesse afin de comprendre. C'est aussi pourquoi je n'arrive pas à faire de la sculpture.

Quand j'y arriverais, ce sera fini et je passerais à autre chose. Malheureusement je crains de ne jamais arrêter de chercher.»

«Quand je sculpte ou dessine un visage, je cherche ce qu'il y a derrière, de générique. Je fais toujours le même visage quel que soit la personne qui me sert de modèle. Si bien qu'à la fin le visage de mon modèle se brouille et devient celui de n'importe qui, de générique. Je ne me rappelle plus de cette personne. Ça peut être embêtant car cela me le fait même avec ma femme.»

«On a toujours pensé, depuis l'art gréco-romain, qu'une tête sculptée comme une tête réelle était le modèle à suivre. À l'inverse, les sculptures "exotiques" (africaines…) nous ont paru primitives. En réalité, c'est l'inverse puisque les sculptures "exotiques" représentent les têtes comme on les voit. En effet un visage de face nous apparaît comme plan et non en trois dimensions. C'est ce que je cherche à faire.»

Alberto Giacometti.

Philémon et le naufragé du A, 1972, Fred.

Une bande dessinée où la poèsie et l'imaginaire permirent à Philémon d'explorer le monde des lettres de l'Océan Atlantique dans lequel il retournera dans chaque album. Elle m'autorisa des rêveries sans fin et un certain goût pour l'exploration…

À Marc,

ta disparition aussi brutale que funeste
déposa sur mon champ de ruine
un terreau d'une richesse sans nom
qu'une vie entière n'aura pas suffit à cultiver

je voudrais célébrer ta mémoire à ma façon
en te dédiant ce travail
et jeter aux orties cette colère sourde
qui habitait le jeune homme que j'étais
et que je me suis efforcé de rester
depuis toutes ces années…

SOMMAIRE

Ø

ÉTUDE THÉORIQUE DU PARTITIONNEMENT DANS LE CADRE DU PARTAGE DE RESSOURCES 82

«Le regard a ses îles, comme il y a des jours et des lunes, qui vous arrivent dans le silence intérieur des mots et les soupirs du souvenir: les yeux partent pour un nouveau voyage qui nous espère, du continent vers les îles choisies, à la rencontre de l'épaisseur de l'espace insulaire. On y saute même quelques vagues avec une mystérieuse et impatiente agilité - vers Elles.

Oui, la curiosité est mobile vers ces alter-egae, nos chères Voisines qui nous font face, nuit et jour et par tous les temps: le grain nocturne les rend à leur discrète magnificence, l'examen de passage peut nous être fatal, nous prenons le risque de passer une nuit blanche à scruter le regard de ces Mystérieuses.

Et le nôtre - notre regard - en quête - y cherche notre histoire commune, révèle notre fraternité, ce goût de sel de la même mer, qui nous apprend la résistante différence, fascinés que nous sommes par leur singularité.

Comment font-Elles?

Le lieu de la beauté recherchée est bien le contact, la rencontre par delà mer et histoire partagées, qui font de ces îles ce que nous sommes, marins dans l'âme, qui cherchons l'ex-île en Elles, comme un répit face à la mer (aux continents?) agitée(s).

Le cri des mouettes est pourtant le même, le rocher - qui a cet Immuable du regard éternel, celui du Sage aux allures éoliennes, le sait: l'appel de chaque côté du rivage est fort, le gué alerte, la volonté minérale, l'attirance séculaire, l'à-venir se nourrira toujours de ce regard croisé anglo-normand.»

Elisabeth du Breil de Pontbriand

INTRODUCTION GÉNÉRALE

I

Après avoir effectué une carrière de chercheur au S.E.P.T.[1] à Caen, puis d'enseignant-chercheur à l'Université de Caen, j'ai éprouvé le besoin, deux ans avant de partir en retraite d'écrire un document rassemblant les travaux de recherche qui m'ont occupé pendant plus de trente années.

Deux jalons principaux (mon doctorat en 1992 et mon Habilitation à Diriger des Recherches - HDR - en 1998) me permirent de baliser plusieurs axes de recherche situés dans le champ de l'informatique, plus précisément dans celui de l'intelligence artificielle distribuée, appelée communément les «systèmes multi-agents», et encore plus précisément sur l'émergence dans les comportements collectifs.

Grâce à la compréhension de mes collègues du département informatique de l'I.U.T. de Caen, que je remercie ici, et conjointement avec les confinements successifs liés au Covid-19, j'ai laissé mes responsabilités administratives pour me consacrer à ce travail.

Il fallait mettre de l'ordre dans un matériau écrit conséquent et des simulations codées, tout aussi conséquentes, afin de raconter un récit possédant un début, un développement et une fin, et, pourquoi pas, des perspectives…

J'ai donc entrepris de développer de nouveaux programmes en langage 'C' (cf. la partie III-C1 du chapitre III), profitant de l'expertise assez pointue que j'avais acquis en programmation système, pendant toutes ces années d'enseignement.

Il s'agissait de mettre au point des programmes les plus performants possibles pour calculer des solutions exactes au problème de partage de ressources C_nS_k (cf. la partie III-A1 du chapitre III), problème central de mon HDR.

Quelque années auparavant (en 1992), je présentais, dans ma thèse de doctorat, une modélisation entièrement spécifiée que j'appelais la *dérive des connaissances* (cf. le chapitre V, partie V-A1).

L'idée était de proposer un modèle gravitationnel, en référence à la physique «classique», qui prendrait en charge les interactions relationnelles entre des connaissances d'une base de connaissances.

Tout ça dans le but de construire un modèle cognitif performant et pertinent pour l'oubli des connaissances, ou plus exactement son organisation efficiente.

Je n'ai jamais eu l'occasion d'implanter ce modèle, mais des collègues d'Avignon l'on fait, et en ont été satisfaits.

Quelques années après cette thèse, je souhaitais trouver un problème concret pour appliquer ce modèle. C'est alors que j'ai travaillé sur le problème du partage de

1 le Service d'Études communes de la Poste et de France Télécom (S.E.P.T.), situé à Caen et créé en septembre 1983 par Louis Mexandeau, alors ministre des P.T.T. sous François Mitterrand et Pierre Mauroy puis Laurent Fabius.

ressources (cf. le chapitre III) en l'attaquant par la face concrète. J'ai commencé à écrire des simulations «multi-agents», dont l'objectif était une résolution autonome et entièrement décentralisée, c.-à-d. sans contrôle central des interactions, ni vues partagées sur des informations lors de la résolution. Je voulais réaliser un programme autopoïétique !

Ce fut le principal travail présenté dans mon HDR.
Cela me fit m'éloigner de la *dérive des connaissances*; je m'enfonçais dans les simulations.

Pour qui a travaillé avec des simulateurs, en particulier dans l'informatique répartie/distribuée, il est facile de comprendre que l'on peut se perdre dans des simulations qui recourent à une quantité de paramètres de plus en plus importante.

Cette période m'a néanmoins sensibilisé aux phénomènes d'émergence dans de tels systèmes. C'est dans cette mouvance que j'ai encadré les thèses d'Hugo Pommier et de Benoît Romito sur le stockage distribué de documents dans des environnements distribués et ouverts, comme Internet.

L'idée initiale de ce travail était de reprendre le côté émergent du comportement des nuées d'oiseaux pour résoudre des problèmes distribués de façon complètement autonome. Grâce aux travaux d'Hugo et de Benoît cette idée initiale, qu'ils ont largement développée, s'est montrée prometteuse, même si elle était un peu en avance sur les technologies des systèmes d'exploitation de l'époque.

En 2015 j'ai été amené à participer aux Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels de Rochebrune, dont le thème était la «singularité». À cette occasion, et en collaboration avec mon collègue Serge Mauger du GREYC (pour la partie linguistique), nous avons écrit l'article intitulé «Singularité et multiplicité, le paradoxe de l'émergence».

Ce fut pour moi l'occasion de proposer une première réflexion sur un cadre théorique permettant de relier la *dérive des connaissances* et la résolution du partage de ressources C_nS_k, en nous appuyant sur la modélisation des lettres de l'alphabet latin.

J'ai mis cet article en intégralité au début du Testament (cf. le chapitre II), car il représente un tournant théorique de mon travail et marque son étendue dans différents domaines des connaissances. C'est une base conceptuelle sur laquelle j'ai travaillé dans les chapitres III et IV.

La réalisation de ce Testament, que je considère davantage comme un nouveau travail avec de réelles avancées et non comme une synthèse, m'a conduit à reprendre le problème du partage de ressources sur le plan théorique (cf. le chapitre III), pour mieux mesurer sa complexité et valider les intuitions proposées dans l'HDR qui alimentent les heuristiques résolutives des simulations au chapitre IV.

J'ai pu ainsi valider ces heuristiques sur des exemples calculables. Je me suis même servi de cette expertise théorique pour affiner les heuristiques.

Le chapitre IV décrit toutes les étapes parcourues (les principes et les réalisations) pour atteindre, avec la version v6 (MVB), des heuristiques qui fonctionnent très bien et résolvent des problèmes incalculables (par la force brute des calculs exhaustifs).

Cette dernière version MVB (cf. la partie IV-E2 du chapitre IV) apparaît, à nos yeux, comme un système autopoïétique capable d'explorer de façon efficiente l'espace des états/solutions du problème C_nS_k, conformément aux résultats théoriques vus au chapitre III.

Ces résultats sont conformes à nos attentes sur les aspects émergents et auto-organisés de tels systèmes, tout en validant expérimentalement notre compréhension théorique du problème C_nS_k.

Au cours de ce travail nous avons entamé une réflexion profonde sur les liens entre *dérive des connaissances* et résolution du partage de ressources via MVB.

Des lectures récentes sur les dernières avancées en matière de cosmologie nous ont permis de développer un cadre théorique plus large que celui proposé au chapitre II sur la singularité.

Ce cadre, que nous appelons «Vers une cosmologie de la connaissance», élargit le champ théorique en permettant de relier clairement nos premiers travaux sur le représentation des connaissances (thèse de 1992) et ceux, plus opérationnels, sur la résolution du partage de ressources (HDR de 1998).

En particulier les travaux sur les liens entre la mécanique quantique (à l'échelle microscopique des particules) et la mécanique «classique» (à l'échelle macroscopique des galaxies et autres corps célestes), nous fournissent un cadre thèorique adapté à notre approche de la représentation des connaissances via les mécanismes d'instanciation, de différenciation et de composition de modèles comme nous l'avons exposé dans la partie V-A2 du chapitre V.

La mise en pratique de ces propositions, dans le cadre de la représentation de connaissances, nécessite de passer à la version MVB^{++} de nos simulations, intégrant justement la *dérive des connaissances*.

L'ensemble du travail conséquent, réalisé pour en arriver à la fin de ce chapitre V, marque pour nous une étape décisive et finale vis-à-vis de notre objectif initial en démarrant ce Testament il y a quatre ans.

Le développement du chapitre VI sur l'instanciation et la différenciation de modèles, dans le cadre de la représentation des connaissances, proposées ici sur l'alphabet latin, nécessiterait de gros développements

dans l'actuelle simulation MVB (passage à MVB^{++}).

Il faudrait :
- introduire la *dérive des connaissances*, mais aussi toute la gestion de la base de connaissances ;
- engrammer les vingt-six lettres de l'alphabet latin en majuscule ;
- reprendre les travaux déjà réalisés pour les lettres 'A' et 'F' (cf. les parties IV-E3/E4 du chapitre IV), avec MVB^{++} et y ajouter les autres lettres ;
- et enfin développer les mécanismes d'instanciation, de différenciation et de composition de modèles dans ce contexte.

Le chapitre VIII fait le lien entre mes activités de recherche qui se terminent avec la rédaction de ce document, et mes activités artistiques que je vais poursuivre maintenant et qui me permettent, dans cet autre cadre, de poursuivre mes expérimentations sur l'émergence de formes, cette fois plastiques et principalement liées à la gestuelle corporelle.

En janvier 2022, j'ai écrit un article intitulé «*Encre-temps*, éloge de l'épaisseur du temps et de la lenteur», pour les Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels de Rochebrune, où j'y avais été invité.

L'expérience artistique, que je mène depuis plusieurs années, et mon activité scientifique se sont rejointes sur un point soulevé dans les conclusions des parties III-D2/D3 du chapitre III.

Il s'agit de l'éloge de la lenteur qui est mathématique dans le chapitre III et artistique dans le chapitre VIII.

Dans les deux cas la lenteur du mouvement, pour le second, et la lenteur du parcours de l'espace des états, pour le premier, se rejoignent pour découvrir des choses intéressantes, que l'absence de lenteur rendrait invisibles.

Nous terminons ce manuscrit par le chapitre IX intitulé «Textes ou la chronologie d'une pensée», qui nous donne l'occasion de partager toutes ces années d'errance dans un brouillard le plus souvent épais, au travers de fragments de notes soulignant les états d'âmes traversés, mais aussi les petites victoires encourageantes et nécessaires pour mener ce genre d'aventure au bout de nos espérances initiales.

J'ai accompagné ces fragments de commentaires, sorte de méta-commentaires, les fragments étant déjà des commentaires sur mon travail en cours.

SINGULARITÉ ET MULTIPLICITÉ, LE PARADOXE DE L'ÉMERGENCE

II

«…En considérant ces "stations" de l'année dynamique, nous sommes frappés de voir qu'elles coïncident avec les principales fêtes de l'année chrétienne. Pâques n'est pas particulièrement mis en évidence, étant une phase du déroulement des saisons et de la lutte entre l'hiver et l'été. Ce n'est pas un événement spécial qui émerge du devenir, — pas un "point singulier". Mais la date de la Pentecôte, fête mobile, dépendant de celle de Pâques, est déjà remarquable: elle se situe toujours dans la période du minimum d'activité atmosphérique qui va du 20 mai au début juin; cette période elle-même ne commence pas à une date précise, comme les autres "stations". Suivent les dates de la St-Michel et de Noël, — car Noël était fixé autrefois au 6 janvier, donc tout près du "point singulier" du 9 janvier, qui marque un maximum de l'activité. Parmi les autres "singularités", qui ont tendance à revenir tous les ans, nommons les fameuses gelées de mai, les "saints de glace", l'été de la St-Martin, les orages de l'Ascension etc…»[1,2].

Théodore Schwenk

1 Extrait du livre de Théodore Schwenk, «Das sensible Chaos/Le Chaos sensible», p.108 dans la deuxième édition française, aux éditions du Centre Triades, 1982.

2 Dans ce texte extrait du livre de Théodore Schwenk, publié en allemand pour la première édition en 1962, on retrouve dans le vocabulaire les notions d'émergence, de singularité par opposition ou en négation à la notion de point singulier dans le sens où un événement n'est pas unique mais revient tous les ans (c'est son côté multiple), ou encore de mobilité; un condensé des ingrédients de notre proposition…

Introduction

II-A

Cet introduction reprend dans sa totalité un article que j'ai écrit en collaboration avec Serge Mauger (laboratoire GREYC, Université de Caen Normandie) dans le cadre des Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels, Rochebrune 2015.

La singularité est une forme canonique émergente, c.-à-d. définie abstraitement dans un processus de couplage autopoïétique, par l'ensemble de formes réelles observables. C'est le paradoxe principal de ce concept, car la chose singulière est une abstraction qui ne prend sens que par l'existence même de formes qui s'en réclament. En d'autres termes, la singularité apparaît avec la diversité, et son émergence nécessite l'existence d'une grande variété de formes. *Être singulier* ne se résume pas à être unique.

La singularité opère par différenciation entre des formes que l'on considère comme voisines. Elles peuvent être en très grand nombre. Cette différenciation introduit une distance limite, par rapport à la *forme canonique*, ce qui permet de ranger telle ou telle forme réelle dans telle ou telle catégorie singulière, représentée par sa forme canonique émergente. Il arrive aussi que certaines formes se situent dans une zone d'incertitude entre plusieurs singularités et ne peuvent être immédiatement caractérisées. Elles sont des non-formes ou des formes en devenir.
Pour illustrer ces notions nous présenterons différents domaines, volontairement très diversifiés, qui nous permettront de suivre quelques pistes de questionnement en partie inédites, sans prétendre à des réponses résolutives[3].
Après nous être appuyés sur les travaux de biologistes cybernéticiens comme F. Varela et H. Maturana (inventeurs, entre autres, des concepts d'autopoïesis et d'énaction) nous interrogerons plus avant la notion de singularité dans son rapport à l'espace public et aux réseaux conçus comme lieux de sociabilité et comme «environnement englobant». Par analogie on peut qualifier ces environnements comme des phénomènes de regroupement qui se présentent comme autant d'avantages dans les stratégies de survie (par exemple les nuées d'oiseaux).

Nous mettrons ensuite l'accent sur la variété des formes des lettres de l'alphabet dans la pratique scripturale, pour montrer comment une lettre, dans la réalité de son exécution graphique, n'est en fait que le rapport à la forme canonique de la singularité émergente correspondante.
Il n'existe pas de 'A' absolu en soi, mais une infinité de représentations tangibles du 'A' qui représente la forme canonique associée. Nous montrerons à partir de cet exemple comment on peut établir un nombre variable de règles qui définissent le 'A' canonique et abstrait.
Nous terminerons en élargissant le propos et en proposant cette définition de la singularité: *la singularité naît, émerge, de la rencontre improbable entre des entités soumises à des contraintes partagées et exogènes, sous l'effet d'un processus de couplage autopoïétique*.

3 En adoptant ce parti pris, qui peut être contestable parce qu'apparemment éclectique, nous avons aussi tenté de nous inscrire dans le fondement même des Journées de Rochebrune et en parfaite conformité avec l'esprit de ce colloque, dont le principe est habituellement d'ouvrir le champ à l'expérience intellectuelle libre, parfois aux limites des investigations purement analytiques. Les pages qui suivent sont donc une exploration large, sinon tous azimuts, du concept de singularité et des associations interdisciplinaires qu'il nous a suggéré. Il ne s'agit rien moins que de suivre ici la remarque que faisait Stefan Zweig à propos de Montaigne : «*Celui qui pense librement pour lui-même honore toute liberté sur terre*».

En paraphrasant Ilya Prigogine dans «*La fin des certitudes*»[4], on dira que la *forme singulière* est une possibilité parmi bien d'autres, qui est actualisée par des évènements. Les lois ne pouvant qu'exprimer des possibilités et non des certitudes.

II-A1) Quelques définitions scientifiques

autopoïèse

ontogénèse

clôture opérationnelle

Autopoïèse vient de «auto» c.-à-d. «soi» et de «poïèse» qui veut dire «création/production». Il s'agit donc de l'autocréation ou création de soi-même. L'*ontogénèse* est le développement d'un individu, en général en s'arrêtant à l'âge adulte. C'est un développement qui s'appuie sur un «couplage structurel» avec l'environnement. La *clôture opérationnelle* est une caractéristique générale de l'autonomie.

En effet, un système autonome est opérationnellement clos si son organisation est caractérisée par des processus :

- dépendant récursivement les uns des autres pour la génération et la réalisation des processus eux-mêmes ;
- constituant le système comme une unité reconnaissable (le domaine) où les processus existent.

énaction

L'*énaction* est un concept riche qui vient compléter les concepts précédents dont l'autopoïèse. Pour Varela et Maturana, qui ont introduit ces concepts, la manipulation symbolique est une description, de niveau supérieur, de propriétés qui se trouvent concrètement matérialisées dans un système distribué et interconnecté sous-jacent.

Le réseau de neurones (celui du connexionnisme) peut donc servir à décrire adéquatement la cognition, mais à condition qu'il puisse produire de la signification.
Et pour qu'un tel réseau puisse produire de la signification, il doit non seulement pouvoir agir sur son environnement, et être sensible à ses variations, mais il doit aussi nécessairement posséder une histoire qui s'inscrit dans le corps et le cerveau de l'organisme.
Car ce qu'on observe concrètement chaque jour, ce sont des agents incarnés qui sont mis en situation d'agir et donc entièrement immergés dans leur perspective particulière. Pour les auteurs, le cognitivisme et les propriétés émergentes du connexionnisme ont le tort de passer sous silence notre expérience humaine quotidienne.

L'énaction vise à embrasser la temporalité de la cognition entendue comme histoire vécue, que cette dernière soit considérée au niveau de l'individu (*ontogénèse*), de l'espèce (*évolution*) ou de structures sociales (*culture*).

Le système «corps-cerveau» contribue à l'avènement conjoint d'un monde et d'une pensée. Pensée qui se constitue à partir de l'histoire des diverses actions accomplies par ce corps dans ce monde.

---

4 Ilya Prigogine, «*La fin des certitudes*», éditions Odile Jacob, collection Sciences, 1996.

Ce qui rappelle les mots du philosophe Jules Lequier qui parlait de «faire et, en faisant, se faire»[5]. On pense également à Montaigne dans un essai intitulé «Du démentir»: «Je n'ai pas plus fait mon livre que mon livre m'a fait, livre consubstantiel à son auteur…».

L'idée derrière le concept d'autopoïèse est de constater qu'avant de pouvoir se reproduire et/ou évoluer, un système vivant doit d'abord être capable de se maintenir en vie de manière autonome…
Une idée que l'on retrouve en arrière plan des travaux de Rodney Brooks (1986). Pour lui l'échec de l'Intelligence Artificielle (IA) repose sur trois choses:

- l'intelligence naturelle repose sur une mobilité dans un environnement dynamique qui n'est pas prise en compte par l'IA classique;
- dans le monde réel il n'y a pas de division entre raisonnement et perception;
- les programmes typiques de l'IA consistent à traiter des symboles élémentaires qui recouvrent en fait des processus complexes.

Brooks proposait le projet de recherche suivant: «*un système intelligent devrait être conçu comme un robot mobile se déplaçant dans un environnement n'ayant pas été spécialement structuré pour lui*». Et Brooks ajoutait: «*le traitement de l'information effectué par ce robot devrait reposer sur une approche distribuée, parallèle et modulaire, en lieu et place de l'approche centralement contrôlée que l'on trouve dans de nombreuses réalisations classiques en IA*[6]».

5 Ce que l'on peut reformuler autrement: «c'est en faisant qu'on fait», ce qui sous-entend le prolongement suivant «et qu'on se fait».

6 Dans «Intelligence naturelle et intelligence artificielle» au PUF, collection «psychologie d'aujourd'hui», p.269, 1993.

7 «…*On situe approximativement la "naissance" de l'espèce humaine à moins 2 800 000 ans, dans cette région d'Afrique proche des grands lacs, cette corne où l'on trouve aujourd'hui la Somalie, le Kenya et l'Éthiopie. La raison pour laquelle je suis pour l'idée d'un polygénétisme des langues humaines, c'est que dès l'apparition de l'espèce, le rift, dont nous voyons encore quelques traces, a eu pour effet de créer une faille qui a entraîné quelque chose de très grave : la niche écologique dans laquelle avait pu naître l'espèce humaine grâce à un certain nombre de facteurs favorables, telle l'abondance des gibiers et des points d'eau, est devenue une niche mortifère. Tout s'est asséché, est devenu invivable, et a été remplacé par une savane à peine arborescente. C'est pour cette raison que l'espèce humaine, à peine née, a dû, pour survivre, quitter ces lieux et se déplacer, parfois très loin. C'est ainsi que la culture dite des galets aménagés, nous la retrouvons dans la vallée du fleuve Jaune, c.-à-d. la région de Pékin…*», p.25-26 dans «*Parler, c'est tricoter*» de Claude Hagège aux éditions de l'aube, 2013.

8 «…*Il y a donc, en ce qui concerne le mouvement de l'eau, deux espèces de vagues bien différentes : dans le ruisseau, la forme de la vague demeure sur place ; mais de l'eau toujours neuve coule à travers cette forme ; dans le lac la forme de la vague progresse sur la surface ; l'eau, par contre, demeure à la même place. Par l'un et l'autre type de vague, l'eau révèle son extrême sensibilité aux pressions extérieures. À la plus petite excitation, provenant d'un caillou dans le ruisseau ou d'une légère brise effleurant le lac, l'eau répond par un mouvement pendulaire. Tout rythme est le résultat de deux facteurs, qui sont ici : 1° l'eau, 2° la force extérieure qu'elle affronte. La vague naît d'une telle composition de forces, par exemple entre l'eau et le vent. Elle surgit à leur point de rencontre. En ceci, l'eau fonctionne un peu comme un organe des sens : elle "perçoit" les chocs les plus infimes et réalise aussitôt, entre les deux antagonistes, un équilibre mobile, rythmique.*», p.25-26 dans «*Das sensible Chaos/Le Chaos sensible*» de Théodore Schwenk, deuxième édition française, aux éditions du Centre Triades, 1982.

9 «…*Une autre caractéristique de la psychologie du nourrisson, plus difficile à saisir, est son incapacité à différencier l'interne et l'externe, le corps propre et les objets externes, soi-même et le monde extérieur. C'est ce que Piaget nomme l'adualisme initial. Cette indifférenciation primitive se double de l'absence de conscience de toute permanence, aussi bien en ce qui concerne les objets externes que les objets internes. L'univers mental du nourrisson est un univers flou et fluide, sans objets au sens où nous l'entendons, tissé seulement des multiples impressions, sensations ou émotions qui l'assaillent. Piaget parle de syncrétisme sensoriel pour désigner cette inorganisation des sensations, cette juxtaposition sans liens des champs sensoriels ni repérages stables dans chacun de ces champs sensoriels. On pourrait dire que, psychiquement, le nourrisson humain est pris dans une sorte de présent permanent, dans lequel émergent ou disparaissent, sans laisser de traces notoires, toutes les impressions, internes et externes mêlées, qui attirent un instant un tant soit peu son attention. Nous pouvons parfois retrouver fugacement cet univers mental, pourtant fort éloigné de nos modes habituels de conscience, par exemple quand nous sommes fortement "désorientés" en nous éveillant soudainement dans un lieu inconnu. Cette notion d'indifférenciation entre l'interne et l'externe est, à vrai dire, une façon adulte assez inadéquate de désigner ce que Piaget nomme égocentrisme initial. Si l'on tente de se placer du point de vue du nourrisson, on ne peut pas dire qu'il y a indifférenciation entre l'interne et l'externe, car cela supposerait de sa part une perception minimale de l'existence d'une telle différence. En réalité, du point de vue du nourrisson,* **TOUT EST INTERNE**. *Le nourrisson est totalement centré, et de façon totalement inconsciente, sur son expérience propre, hors de laquelle rien n'existe pour lui. Dans cet adualisme initial, seul existe ce qui existe à la fois pour soi et en soi, aux sens littéraux de ces expressions. Piaget reprend ici l'idée psychanalytique de narcissisme primaire, mais en soulignant qu'il s'agit d'un "narcissisme sans Narcisse" du fait de son inconscience totale. Du fait de l'absence de toute intuition de l'existence d'un univers hors de soi, "objectal", c'est un narcissisme qui ne se différencie de rien et ne s'oppose à rien…*» dans «*Quelques repères en psychologie de l'enfant et du préadolescent*» de Daniel Calin, http://dcalin.fr/cerpe/cerpe09.html.

10 «…*Notre proposition est que les êtres vivants sont caractérisés par le fait que, littéralement, ils sont continuellement en train de s'autoproduire. Nous nous référons à ce processus lorsque nous appelons l'organisation qui les définit l'organisation autopoïétique. Cette organisation repose sur des relations plus faciles à mettre en évidence au niveau cellulaire. Tout d'abord, les composants moléculaires d'une unité autopoïétique cellulaire doivent être reliés dynamiquement dans un réseau d'interactions continues. Aujourd'hui on connaît un grand nombre de transformations chimiques spécifiques de ce réseau, que les biochimistes regroupent sous l'expression de métabolisme cellulaire… Il est intéressant de noter que ce métabolisme cellulaire produit des composants qui font partie du réseau de transformations qui les a produits. Certains de ces composants forment une frontière, une limite à ce réseau de transformations. En termes morphologiques, la structure qui permet ce clivage spatial est appelée une membrane. Cette frontière membranaire n'est pas un produit du métabolisme cellulaire comme le tissu est le produit d'une machine à produire du tissu. La raison en est que cette membrane ne se contente pas de limiter l'extension du réseau de transformations qui a produit ses propres composants, mais qu'elle participe elle-même à ce réseau. S'il ne bénéficiait pas de cet arrangement spatial, le métabolisme cellulaire se désintégrerait en une soupe moléculaire qui se répandrait tout autour et ne constituerait plus l'unité discrète qu'est la cellule. Ce que nous avons est donc une situation unique en ce qui concerne les transformations chimiques : d'une part, on voit un réseau de transformations dynamiques qui produit ses propres composants et qui est essentiel pour élaborer une frontière ; d'autre part, on voit une frontière qui est essentielle pour l'opération des transformations du réseau qui en font une unité… Un des aspects les plus frappants des systèmes autopoïétiques est qu'ils démarrent d'eux-mêmes et deviennent distincts*

II-A2) Mobilité cognitive, où le principe du « pas de côté »

La vie repose sur un certain nombre de principes tels que la reproduction, l'adaptation et l'évolution. Si l'on pense à la reproduction, très vite la notion de mobilité apparaît comme un mécanisme sous-tendant une reproduction viable[7]. L'espèce humaine ne peut se reproduire de façon consanguine sans risquer la dégénérescence. C'est bien d'une première forme de mobilité qu'il s'agit. Comme nous allons le voir la mobilité est multiforme, elle intervient en permanence, partout dans les systèmes vivants ou non vivants[8].

L'adaptation suppose qu'un système vivant évolue dans un environnement qui peut être changeant. Chaque individu apparaît alors comme faisant à la fois partie de cet environnement, tout en lui étant extérieur[9]. Participant à cet environnement, lui-même évolue, d'où le troisième grand principe d'évolution des systèmes vivants.

Cette circularité bien connue des systèmes vivants, modifiant l'environnement dans lequel ils évoluent, qui à son tour modifie les systèmes vivants eux-mêmes, et qui pourrait prendre son origine symbolique et populaire dans le paradoxe de « l'œuf et de la poule… », déjà étudié par Aristote, mais plus récemment par Humberto R. Maturana et Francisco J. Varela à travers leur concept d'organisation autopoïétique[10], là encore se fonde sur cette propriété fondamentale qu'est la mobilité.

Aussi loin que l'on remonte dans nos connaissances sur l'origine de l'univers, la mécanique céleste a produit juste après le Big Bang un territoire gazeux, donnant naissance à la matière, mais aussi à des molécules organiques, typiques des organismes cellulaires modernes[11]. Ce territoire n'a cessé depuis d'évoluer. C'est un système dynamique avec ses propres règles, en expansion, berceau de l'apparition de la vie. Son étendue donne aux systèmes vivants qui le peuplent, l'impérieuse nécessité d'un déploiement perpétuel leur assurant leur survie, avec d'autant plus de force que ce dit territoire obéit à des règles qui peuvent apparaître comme hostiles à la vie.

La vie est placée dans un territoire qu'elle doit apprendre à connaître, à domestiquer pour survivre, c.-à-d. vivre sur le territoire. Elle doit évoluer si ce territoire évolue, et s'adapter à ce dernier. Il est inutile de balayer les trop nombreux exemples que nous connaissons en matière d'adaptation des espèces vivantes (humaines, animales, cellulaires,…). Dans tous les cas la mobilité joue un rôle central. Citons l'exemple de certaines plantes qui se regroupent en buisson pour mieux résister aux conditions environnementales difficiles ou encore celui de la stratégie de survie de la douve du foie de veau, qui utilise les herbes hautes et les fourmis comme vecteur de mobilité pour atteindre sa cible. La mobilité permet à l'homme d'évoluer en harmonie avec son environnement, rencontrer d'autres personnes, visiter des pays, des cultures différentes, s'enrichir de la différence. Au XXI^{e} siècle il a développé de nombreux outils, techniques

de leur environnement à travers leur propre dynamique, si bien que l'un et l'autre sont inséparables…» ,dans «*L'Arbre de la connaissance. Racines biologiques de la compréhension humaine*», p.32-35, aux éditions Addison-Wesley, 1992.

11 S. L. Miller a réalisé en 1953 des expériences en laboratoire montrant qu'en prenant un échantillon de l'atmosphère primitive et en lui fournissant de l'énergie de façon adéquate on obtient des molécules organiques similaires en complexité à celles présentes aujourd'hui chez les êtres vivants.

12 «*…parmi l'ensemble des informations sensorielles qui lui parviennent, le système nerveux central (SNC) doit sélectionner celles qui sont pertinentes, puis les stocker sous une forme définitive. La sélection de certaines informations nécessite, d'une part, de pouvoir stocker temporairement un maximum d'informations afin de réaliser des associations éventuelles avec des évènements ultérieurs et, d'autre part, de pouvoir rappeler des informations déjà stockées de façon à les comparer aux informations nouvelles… L'information transite tout d'abord par une mémoire sensorielle dont la capacité est très limitée et dont la durée est de l'ordre de la seconde. Elle transite ensuite vers un stock à court terme, contenant en moyenne sept items renouvelés en permanence et dont la durée de stockage n'excéderait pas une à deux minutes. L'information est enfin transférée dans un stock à long terme où elle est conservée sous une forme stable et définitive constituant alors l'engramme. Il faut noter que cette dernière étape n'est pas systématique. Chacun sait en effet que toutes les informations reçues ne sont pas stockées à long terme. Pour que ce passage s'effectue, il faut soit que l'information qui sous-tend l'apprentissage soit répétée plusieurs fois, soit qu'elle ait une forte valeur affective. Une hypothèse, classiquement admise, considère que les trois stocks mnésiques sont organisés en série, chacun des stocks alimentant le suivant…*», dans «*Neurophysiologie, organisation et fonctionnement du système nerveux*», D. Richard, D. Orsal aux éditions Dunod, p.477.

13 Dans son livre «*La sculpture du vivant. Le suicide cellulaire ou la mort créatrice*» aux éditions du Seuil dans la collection Points/Sciences, 2003, Jean-Claude Ameisen développe l'importance de la mobilité dans l'existence de la vie et plus généralement de la complexité. «*…De la diversité naît la complémentarité, de la complémentarité, l'interdépendance et de l'interdépendance, la complexité. Des familles cellulaires différentes (foie, cœur…) se regroupent dans des régions distinctes, qui se côtoient, découpant la société cellulaire en de grands territoires. Émerge l'ébauche d'un corps constitué de régions, de pôles, de frontières, un haut et un bas, un avant et un arrière, une droite et une gauche. Des cellules se déplacent, voyagent d'un territoire à l'autre, acquièrent au contact d'autres cellules des propriétés nouvelles. La cellule-œuf a donné naissance à un univers multiple et complexe qui se régionalise, se segmente et s'organise en modules qui s'emboîtent les uns dans les autres et s'intègrent pour former un individu. La potentialité de construire un corps dépend désormais des interactions de plus en plus nombreuses entre les différentes familles cellulaires issues de cette cellule originelle. Les tissus et les organes s'auto-organisent à partir de leurs ébauches. La société cellulaire est devenue l'architecte de sa propre construction. Ainsi, la métamorphose d'une cellule-œuf en embryon implique trois grands phénomènes complémentaires : la division ou le dédoublement cellulaires, qui donne naissance à la multitude ; la différenciation cellulaire, qui crée l'asymétrie, la diversité, la régionalisation et la complémentarité ; et la migration – le déplacement des cellules à travers le corps – qui permet de répartir et de recomposer dans l'espace cette diversité. Mais à ces trois phénomènes s'en ajoute un autre, étrange, et apparemment paradoxal, la mort cellulaire…*»

14 «*…nous pénétrons dans la fondation lorsque, changeant d'espace, nous atteignons un terrain sans repère de distance. De même que la topologie fonde les espaces métriques, de même en la culture gît l'infrastructure des constructions humaines…*» dans «*L'Incandescent*» p.146, Michel Serres, aux éditions le Pommier, 2003.

15 L'une des plus célèbres conjectures qui a tenu le monde scientifique en haleine pendant trois-cent-cinquante-huit ans fut la démonstration du dernier théorème de Fermat par Andrew Wiles. «*…Wiles compare son expérience à l'exploration d'une grande maison inconnue. "On entre dans la première chambre et elle est obscure. On se heurte aux meubles, on finit par connaître leur emplacement. Après quelques six mois, on finit par trouver le commutateur et soudain la pièce est éclairée. On peut voir exactement où l'on se trouve. Puis on passe à la pièce suivante, et l'on affronte à nouveau six mois d'obscurité…"*», dans «*Le dernier théorème de Fermat*», p.252, Simon Singh, aux éditions Pluriel, 2010.

16 «*…Au cours du processus de création, l'artiste passe alternativement du point de vue du peintre à celui qui regarde la toile… Au moment où l'artiste observe l'œuvre qui est en train de se réaliser sous sa main, un autre œil observe l'artiste lui-même. Quand un artiste filme ses gestes quotidiens, s'il reste un artiste, c'est parce qu'il possède ce troisième œil et qu'il a, au préalable, imaginé cette observation de lui-même afin de se considérer comme une œuvre d'art. Mais ce n'est pas la caméra vidéo qui observe l'artiste : si ce troisième œil n'est que la projection de son narcissisme, l'image qui en résultera ne pourra être que l'image de lui-même et non pas son art. Lorsque ce troisième œil dépasse le narcissisme de l'artiste, lorsqu'il en est relativement indépendant, autrement dit quand une certaine distance s'installe, un refroidissement de l'ardeur narcissique se produit, et l'artiste commence à regarder calmement l'œuvre qu'il est en train de réaliser. Son regard froid évalue le travail, le juge, fait le tri et décide. C'est sous le regard de ce troisième œil que l'artiste crée, dépassant ainsi la création seulement artisanale…*», dans «*De la création*», p.210-211, Gao Xingjian aux éditions du Seuil, 2013. Un autre exemple important est rapporté par Aragon sur Matisse dans «*Aragon, Henri Matisse,*

et pratiques nouvelles de communication, d'échange et de partage. De nouvelles formes de mobilités apparaissent puisqu'à défaut d'aller physiquement à l'autre bout du monde, il fait venir le monde à lui. La mobilité spatiale devient spatio-atemporelle. Les livres et autres traces mémorielles permettent un voyage dans le passé, voire dans le futur imaginé, alors que les outils de communication permettent un voyage dans l'instant, dans le présent. C'est l'instantanéité des choses. L'instant présent devient à lui seul un espace de mobilité.

Partout où se développe la vie la mobilité demeure un principe incontournable. Pourtant il est des domaines où cela ne saute pas aux yeux; c'est le cas du cerveau humain. En fait il faut parler du système nerveu et neuronal de l'homme. Chaque être humain abrite une complexité en nombre de composants et d'interactions, avec ses cent milliards de neurones et ses dix-mille connections (en moyenne) depuis chaque neurone vers les autres. Elle est à l'échelle de l'univers intergalactique avec ses cent milliards (environ) de galaxies, réparties dans un univers observable dont le diamètre est estimé à cent milliards d'années-lumière. Nous pouvons parler d'une complexité fractale entre l'univers et les humains.

Tous ces neurones sont répartis dans tout le corps humain jusqu'aux extrémités des membres, même si leur grande majorité se trouve dans le cerveau. La structure des connexions dans le cerveau nécessite une organisation hautement répartie avec des zones de traitement, d'échange et de transmission de l'information pour lesquelles la mobilité intervient[12,13]. Nous avons une connaissance assez précise du fonctionnement des circuits neuronaux. L'information transite sous forme électrique à travers des liaisons neuronales dans des gaines conductrices. Lors du passage d'un neurone à un autre il y a une rupture de ces gaines conductrices au niveau du dendrite, entraînant une discontinuité électrique dans la transmission de l'information. C'est la chimie qui prend alors le relais pour atteindre la synapse du neurone connecté.

Arrivé à ce point, une confusion pourrait apparaître assez naturellement entre mobilité et échange d'information, voire communication. La mobilité utilise la communication tout en apportant autre chose. De la même façon la cognition utilise la mobilité. Bien que nous connaissions avec beaucoup de précisions le fonctionnement physico-chimique de la transmission de l'information dans le système neuronal, la production des symboles reste un grand mystère. Rendre mobile une entité quelle qu'elle soit dans un environnement donné, revient a lui donner la possibilité de changer son point de vue, faire un «pas de côté».
Dans un environnement physique le «pas de côté» peut être dans les trois dimensions, dans un espace abstrait il peut prendre d'autres formes[14]. Cela peut être un déplacement dans le temps, un changement de versions, le point de vue d'un autre… On peut évoluer simultanément dans plusieurs environnements (physique, mental…) et ne se déplacer que dans un seul ou dans plusieurs d'entres eux.

roman», p.132, aux éditions Gallimard, dans la collection Quarto, 1998. «*…Un des grands mystères matissiens… c'est ce double jeu du modèle : qu'il ne puisse se passer d'un modèle d'une part, et que le modèle inspire d'autre part quelque chose de si libéré de lui… C'est bien par là que nous sommes au-delà du portrait, dans cette pièce pavoisée à la dissemblance d'un même visage, d'un même modèle. Cent fois je pourrais répéter ce vers que j'aime tant : "Amie éclatante et brune", comme cent fois Matisse a repris le visage semblable et toujours différent…*» Le pas de côté de Matisse est l'un des plus ténu qui soit, mais d'une très grande importance. Il n'est qu'à voir, dans «*Henri Matisse, Traits essentiels. Gravures et monotypes 1906-1952*», aux éditions Cabinet des estampes, Genève, 2006, ses séries de portraits où le trait unique noir posé sur la feuille blanche, ou blanc sur la feuille noire, modifie le rendu du portrait. Matisse a cherché à travers ses dessins à rendre un trait d'une simplicité confondante porteur d'une émotion subtile.

17 «*…L'aléatoire est un associé, mais il n'est pas le patron. Je m'arrange pour qu'il puisse me faire des propositions. Je mets au point des stratégies qui lui permettent d'en faire. Mais c'est toujours moi qui décide en définitive. J'accepte ou je refuse ses propositions, j'évalue intuitivement leurs limites, si elles ont un avenir ou non…*», dans «*Soulages, les peintures 1946 - 2006*», p.165, de Pierre Encrevé aux éditions du Seuil. C'est une autre façon d'exprimer le couplage autopoïétique ici entre l'artiste et son environnement de création. Soulages décrit ici ce que Prigogine énonce comme l'expression de possibilités et non de certitudes. On retrouve également une référence «au troisième œil» de Gao Xingjian (cf. note de bas de page précédente) et donc directement au principe du «pas de côté».

18 «*…La peinture moderne a tenté… de chercher un traitement nouveau de l'espace. Cézanne et Picasso se sont débarrassés de la perspective propre à la géométrie euclidienne ; ils ont supprimé la profondeur et fondés le cubisme…*», p.*213*, «*…La peinture impressionniste est plus éclatante que les couleurs vues à la lumière naturelle, en réalité elle présente déjà des images visuelles intérieures et non une imitation de la nature… les contrastes violents entre couleurs deviennent le langage dominant de la peinture… Ces paysages intérieurs légèrement troubles, espaces irréels fugaces aux couleurs sombres insaisissables, s'approchent du noir et blanc, ces images évoluent en un clin d'œil comme dans une scène de rêve… Comment alors trouver un langage plastique plus proche de ces images intérieures ? Dans cette ambiguïté intermédiaire entre abstrait et figuratif se cache justement un grand portail dissimulé qui, une fois ouvert, est d'une profondeur insondable… La dénotation et l'allusion sont les clefs qui ouvrent ce portail. Il y a deux types de méthode dans la tradition des arts plastiques : la représentation et l'expression. La première imite la relation naturelle entre les formes, la lumière, les couleurs et l'espace… L'expression en revanche a recours aux sentiments subjectifs et au partage ou à l'épanchement des sentiments… La dénotation en revanche peut devenir une telle méthode si elle est utilisée pour défricher un nouveau domaine plastique entre l'abstrait et le figuratif… Ce domaine a été révélé pour la première fois par… Turner, dont les paysages dans la brume peuvent être rattachés à l'impressionnisme, mais aussi à l'abstraction… l'espace et la profondeur de champ sont plus diffus, la source de lumière n'est pas claire, les images et les sentiments se mélangent…*», dans «*De la création*», p.151-152, Gao Xingjian aux éditions du Seuil, 2013.

19 «*…Le rythme linguistique ou grammatical est défini en français par la place d'un accent de groupe : celui-ci frappe la dernière syllabe non muette d'un groupe syntaxique — des cheveux blonds/un clair ruisseau/un endroit tranquille — La distribution des accents détermine le rythme : temps marqué, temps non marqué pouvant être plus ou moins régulièrement espacé. L'accent n'est donc pas lexical, il est mobile et dépend de l'organisation syntaxique des textes…*», dans «*Analyses stylistiques. Formes et genres*», Catherine Fromilhague et Anne Sancier-Chateau aux éditions Nathan Université, 2000.

20 Avec «*Shitao. Les propos sur la peinture du moine Citrouille-amère*» aux éditions Hermann, 1996, écrit aux environs de 1710 par le peintre Shitao, nous avons là un traité qui possède une double lecture. La première concerne un manuel de peinture où Shitao développe le concept d'*Unique Trait de Pinceau*, mais elle se double d'une deuxième lecture philosophique, qui, au dire des critiques chinois, représente une des expressions les plus hautes et les plus complètes de la pensée esthétique chinoise. On retrouve très clairement le parallèle entre les principes philosophiques énoncés par Shitao dans son traité et la notion de singularité, telle que nous allons la développer plus loin. P.18, «*…Shitao ramène la peinture à sa forme la plus élémentaire et la plus humble : un simple trait de pinceau ; mais un simple trait de pinceau est aussi l'Unique Trait de Pinceau, mesure universelle de l'infinité des formes, commun dénominateur et clé de toute création…*». On trouve dans l'*Unique Trait de Pinceau* la singularité des singularités, celle à partir de laquelle toute forme plastique (des formes les plus hautes de la peinture que sont les paysages, aux catégories inférieures que sont les personnages, les animaux et les architectures) peut être instanciée, peut apparaître, la matrice de toute représentation. Plus qu'un simple savoir-faire, «*…à la portée de n'importe quel artisan consciencieux, capable de reproduire l'évidence objective de ses sens…*», «*…peindre est difficile avant de peindre. Le premier travail du peintre est de développer en lui cette source intérieure du cœur… ; l'exécution matérielle de la peinture ne pose alors plus de problème, elle n'est que la conséquence naturelle et aisée de cette vision spirituelle qui la précède…*» Cette forme particulière de couplage autopoïétique conduit à cette singularité exceptionnelle que l'on retrouve dans tous les domaines artistiques (musique, peinture,…) et que partagent les très grands artistes. Elle leur permet de réaliser des choses splendides et complexes avec une facilité apparente qui laisse le commun des mortels dans une très grande perplexité.

Dans le système neuronal, la mobilité intervient dans la mobilisation d'une partie des neurones pour répondre à un stimuli.
La communication entre les neurones et les synapses, suivant des protocoles déterminés, via les synapses et les dendrites, permet de mettre en action telle ou telle région (circuit) du système neuronal. Dans ce cas la mobilité est multimodale car elle peut être électrique ou chimique.
La mobilité permet le «pas de côté» nécessaire à une sollicitation externe, tout comme la table de routage (protocole IP dans les réseaux informatiques) pour acheminer des paquets de données d'un point à un autre.

Les activités intellectuelles et sensibles n'échappent pas à ce principe. C'est le cas en recherche scientifique pour résoudre une conjecture[15]; cela nécessite des méthodes, des outils ou encore de se déplacer dans l'espace des solutions.
C'est aussi le cas en art[16,17,18], en littérature[19], en philosophie[20],…

Qu'est-ce que la singularité, sens commun, approche sensible ?

II-B

21 Les travaux de René Thom sur sa « *théorie des catastrophes* » montrent qu'en tout point singulier, au sens mathématique du terme, apparaît dans son voisinage immédiat des classes d'universalité au nombre de sept, qui peuvent être qualifiées d'autant de singularités différentes. On peut définir une classe d'universalité par le fait que certaines grandeurs se retrouvent à l'intérieur de tous les éléments d'une même classe. Autrement dit tout se passe de la même manière dans le voisinage de la singularité d'une classe donnée.

II-B1) Définitions communes

Wikipédia : La singularité décrit le caractère singulier de quelque chose ou de quelqu'un, en particulier :

- *en physique* : la singularité gravitationnelle est le point particulier de l'espace-temps au voisinage duquel certaines quantités décrivant le champ gravitationnel deviennent infinies ;
- *en mathématique* : c'est le point où un objet mathématique n'est pas bien défini, par exemple une valeur où une fonction d'une variable réelle devient infinie ou encore un point où une courbe a plusieurs tangentes ;
- *en technologie/futurologie* : c'est la rupture à partir de laquelle la civilisation humaine connaîtra une croissance technologique d'un ordre supérieur.

Larousse :

- *littéraire* : caractère de ce qui est unique en son genre, la singularité de chaque vie ;
- caractère original ou étrange, insolite de quelque chose : « la singularité de sa tenue l'a fait remarquer » ;
- acte qui révèle ce caractère : ses singularités n'étonnent plus personne ;
- trait distinctif de la catégorie du nombre indiquant la représentation d'une seule entité isolable ;
- particularité survenant en un point singulier d'une courbe ou d'une surface[21].

Il est intéressant de noter le caractère autoréférent de la définition d'une singularité (Larousse) dans le domaine mathématique des courbes et des surfaces. Dans les définitions plus précises qu'en fait Wikipédia, on parle d'objet pas bien défini, d'une variable infinie de quantités infinies… Les définitions (Larousse) du sens commun se réfèrent davantage à la notion d'unicité, sans être, comme nous le verrons par la suite, en contradiction ou en désaccord avec le paradoxe de la singularité. La quatrième définition proposée par le Larousse est intéressante car elle fait apparaître explicitement les notions de catégorie, de traits distinctifs, la notion de représentation et enfin, et non des moindres, la notion d'entité isolable. Nous avons dans cette définition tous les ingrédients du processus d'émergence générateur de singularités.

II-B2) Références en littérature

En littérature les exemples sont nombreux et fort instructifs pour nous éclairer sur ce qui se cache derrière cette notion d'unicité trop souvent confondue un peu vite avec la notion de singularité.
En réalité plusieurs niveaux sont aplatis, masquant ainsi les processus complexes de constitution nécessaires à l'apparition de singularités. Comme nous allons le voir sur des exemples précis, une singularité est un élément unique, une forme canonique émergente issue de processus complexes, basés entre autres choses sur la notion de couplage autopoïétique. Cette entité abstraite est directement utilisable dans des niveaux d'abstraction

22 Dans « *Le Pic de la Mirandole du XIX^e siècle* » par Michel Mourlet, p.167 dans « *Littré au XXI^e siècle. Le colloque du bicentenaire* » aux éditions France Univers, 2003.

23 Aux éditions Droz, dans la collection « *Histoire des idées et critique littéraire* », 2008.

24 Nathalie Heinich, « *L'élite artiste. Excellence et singularité en régime démocratique* », Paris, éditions Gallimard dans la collection « *Bibliothèque des sciences humaines* », p.172, 2005.

25 *J'entendrai ce terme d'instance dans la définition qu'en donne Jacques Dubois d'un « rouage institutionnel remplissant une fonction spécifique dans l'élaboration, la définition ou la légitimation d'une œuvre »*; Jacques Dubois dans « *L'institution de la littérature. Introduction à une sociologie (1978)* », Bruxelles, aux éditions Labor-Nathan dans la collection « *Dossiers Média* », 1986, p.82.

supérieurs. Elle se définit continuellement, grâce à de nombreuses entités (ses instances constitutives) qui partagent certaines caractéristiques.

«*…Ces motifs qui militent en faveur du Littré, de sa valeur permanente, de son actualité toujours fraîche, ne sont rien encore. Sa plus étonnante vertu (qu'il partage, il faut le reconnaître, avec le Grand Larousse du XIX*e *siècle) a été parfaitement mise en lumière par l'avertissement qui ouvre l'édition de Monte-Carlo: "Le Littré offre la singularité d'être aussi un ouvrage de lecture courante quasi inépuisable."*»[22]
Il faut remarquer l'aparté de ce passage. L'auteur montre entre les lignes (aparté) les deux niveaux constitutifs de la singularité que représente le Littré. Nous avons le dictionnaire lui-même, instance d'une entité, au même titre que le grand Larousse, et la (ou les) propriété(s) principale(s) qui en découle(nt), qui caractérise(nt) l'élément canonique émergent unique…
À propos de la définition du cénacle par Émile Littré, commentée par Anthony Gliner p.16 dans «*La querelle de la camaraderie littéraire. Les romantiques face à leurs contemporains.*»[23]
«*…Celle d'Émile Littré ensuite: "Réunions d'hommes de lettres, d'artistes, etc, qui se voient souvent et sont accusés de s'admirer mutuellement". La première proposition met l'accent sur la communauté de pensée, de vie et de visée entre les membres du cénacle, la seconde sur les effets pervers engendrés par les rencontres régulières. À leur intersection, dans la tension entre une pratique sociale et les discours que cette pratique fait surgir, réside sans doute le problème qu'a posé le phénomène cénaculaire au XIX*e *siècle. Osons alors quelques éléments de réponse au-delà de ces définitions: un cénacle est, à première vue, une assemblée d'écrivains et d'artistes réunis dans un espace privé ou semi-privé, à l'abri du passage et fermé des auditeurs libres. À la différence d'une association, cette assemblée n'existe par nulle autre institutionnalisation qu'elle-même, par la simple agrégation (le cénacle est donc à la fois un réseau et un lieu de sociabilité). Mais le cénacle n'est pas que cela: il forme dans l'espace public une entité humaine spécifique, fondée sur une certaine homogénéité et une certaine cohésion. Doté dune "singularité collective"*[24]*, il vaut plus que la somme des parties, de sorte que c'est en tant que communauté d'intérêt qu'il est susceptible d'influer sur la configuration générale du champ. Le cénacle est donc une instance*[25]*. Enfin, le cénacle est intimement associé, c'est sa raison d'être sociologique, à un ensemble de valeurs esthétiques communes à ses membres, dont il vise, avec plus ou moins de fortune, la systématisation et l'explication (le cénacle est donc l'habitacle d'un mouvement littéraire et/ou artistique). Balloté entre ces identités sociales différentes mais complémentaires, le cénacle se caractérise encore par l'homogénéité professionnelle et sociale de ses membres — on s'y réunit entre pairs — et par une intense cohésion interne sous-tendue par des relations dilectives, graduées depuis la fraternité de circonstance jusqu'à l'amitié indéfectible. Cet ensemble de traits distinctifs, marqué du sceau du*

26 « *Systèmes d'Information Ouverts : Sémantique Interactionnelle des Connaissances et Systèmes Multi-Agents* », François Bourdon, rapport d'HDR de l'Université de Caen, spécialité Informatique, 1998.

27 « *Un modèle de dérive des connaissances. Application en bureautique* », *François Bourdon*, thèse de Doctorat de l'Université du Maine, spécialité Intelligence Artificielle, 1992.

paradoxe et de l'instabilité (comment être à la fois communauté d'intérêt et regroupement fondé sur des formes aigües d'amitié et d'admiration?), ne permettent pas, contrairement à une opération répandue depuis que Sainte-Beuve s'est écrié, sous le coup du dépit: "Le Cénacle n'était après tout qu'un salon", de confondre la forme-cénacle avec la forme-salon…»

Ce texte appelle de très nombreux commentaires. «*Le cénacle forme dans l'espace public une entité humaine spécifique*», (émergence), «*fondée sur une certaine homogénéité et une certaine cohésion. Doté d'une singularité collective…*» (apparition d'un deuxième niveau). C'est la définition même de la singularité. La singularité n'est pas le ou les caractères/traits uniques et partagés par plusieurs entités, mais bien l'entité émergente, qui se caractérise par le ou les caractères «uniques» partagés. Elle devient autonome et réutilisable dans des niveaux de représentation supérieurs. «*…Il vaut plus que la somme de ses parties…*». On voit ici une référence implicite à l'émergence d'une forme canonique. «*…de sorte que c'est en tant que communauté d'intérêt qu'il (le cénacle) est susceptible d'influer sur la configuration générale du champ…*» On voit ici poindre l'autonomie attenante à cette forme canonique émergente. «*…À la différence d'une association, cette assemblée*» (forme canonique émergente) «*n'existe par nulle autre institutionnalisation qu'elle-même, par la simple agrégation*» (dynamicité, auto-organisation…) «*(le cénacle est donc à la fois un réseau et un lieu de sociabilité)…*» Cette conclusion, ici entre parenthèses, mentionne explicitement la dualité du cénacle: un réseau et un lieu de sociabilité. On retrouve les deux grandes caractéristiques de l'émergence à savoir des interactions et un environnement. On peut faire ici un parallèle avec les travaux[26] sur l'émergence et la représentation des connaissances, dont nous parlerons en fin d'article, dans lesquels le modèle d'encodage proposé, appelé C/S, représente l'environnement d'engrammage des connaissances, constitutif du premier niveau de singularité; auquel on peut ajouter le modèle relationnel[27], celui des interactions, constitutif des niveaux supérieurs de singularité.

Le 16 octobre 1960, le critique d'art français Pierre Restany publie à Paris et Milan le premier «*Manifeste du Nouveau Réalisme*». Le 27 octobre 1960, au domicile d'Yves Klein, Arman, François Dufrêne, Raymond Hains, Martial Raysse, Daniel Spoerri, Jean Tinguely et Jacques Villeglé signent en neuf exemplaires la déclaration constitutive du groupe des Nouveaux Réalistes, affirmant par là leur «*singularité collective*».

Du singulier multiple à la singularité : le paradoxe.

II-C

28 « *…Au cours de l'histoire de la Terre, c'est seulement lorsque les conditions furent réunies pour la formation de molécules organiques telles que les protéines, qui ont une complexité et une flexibilité énormes, que les conditions étaient aussi rassemblées pour la formation d'unités autopoïétiques. En fait, on peut supposer que lorsque toutes ces conditions suffisantes furent réunies dans l'histoire de la Terre, les systèmes autopoïétiques se formèrent inévitablement. Ce moment marque le point auquel on peut se référer pour l'origine de la vie. Cela ne veut pas dire qu'il se produisit une seule fois et en un seul endroit, pas plus qu'il ne nous est possible de lui donner une date précise. Toutes les évidences disponibles nous confortent dans la croyance qu'une fois que les conditions furent réunies pour l'origine des systèmes vivants, ceux-ci sont apparus de nombreuses fois. C.-à-d. que de nombreuses unités autopoïétiques avec de nombreuses variantes structurales ont émergé en de nombreux endroits sur la Terre, au cours d'une période qui a pu couvrir plusieurs millions d'années.* » dans « *L'Arbre de la connaissance. Racines biologiques de la compréhension humaine* », p.39-40, Humberto R. Maturana, Francisco J. Varela aux éditions Addison-Wesley, 1994.

29 « *…les unités autopoïétiques spécifient la phénoménologie biologique comme phénoménologie propre à ces unités possédant des traits distincts de la phénoménologie physique. Il en est ainsi, non parce que les unités autopoïétiques vont à l'encontre de certains aspects de la phénoménologie physique – puisque leurs composants moléculaires doivent suivre toutes les lois de la physique – mais parce que les phénomènes que ces unités génèrent en fonctionnant comme unités autopoïétiques dépendent de leur organisation et de la façon dont cette organisation se met en place, et non de la nature physique de leur composants (qui détermine seulement leur espace d'existence).* », dans « *L'Arbre de la connaissance. Racines biologiques de la compréhension humaine* », p.40-41, Humberto R. Maturana, Francisco J. Varela aux éditions Addison-Wesley, 1994.

30 Un exemple spectaculaire est celui des ponts de fourmis pour franchir un obstacle.

31 Ces travaux ont montré la pertinence de cette métaphore dans le domaine d'Internet, tout en pointant du doigt les manquements des ordinateurs actuels à gérer simultanément de très grandes quantités d'agents logiciels autonomes. Ils ont été réalisés par Hugo Pommier et Benoît Romito : Hugo Pommier, « *Placement et stockage de l'information bio-inspiré : une approche orientée agent mobile.* », thèse de doctorat de l'Université de Caen Basse-Normandie, en Informatique, 2010.

Benoît Romito, « *Stockage décentralisé adaptatif : autonomie et mobilité des données dans les réseaux pair-à-pair.* », thèse de doctorat de l'Université de Caen Basse-Normandie en Informatique, 2012.

Depuis l'origine de la vie sur terre, et probablement ailleurs si elle existe, des mécanismes intrinsèques à la vie se sont manifestés. Avant l'apparition de la vie il y avait un univers avec ses propres contraintes, sa propre dynamique, formant ainsi un environnement autonome et indépendant de la vie. Pour que la vie apparaisse il a fallu que des conditions particulières soient réunies, au-delà même d'avoir, à un moment donné, tous les ingrédients nécessaires à cette apparition. Trivialement on retrouve cette idée pour monter une mayonnaise ou allumer un feu de bois dans une cheminée. On peut penser que de nombreuses tentatives ont échoué avant que la vie ne survive[28]. La vie n'existe donc que si elle perdure, cela fait partie de sa constitution/définition. Depuis le départ, la vie a du explorer l'environnement dans lequel elle était contrainte d'évoluer, s'y déplacer, y faire son «*pas de côté*». Cette propriété lui reste fondamentalement ancrée encore aujourd'hui.
Pour tenter d'expliquer le fait que la mayonnaise prenne — nous voulons dire que la vie survive — nous allons réinvestir la notion intuitive de couplage autopoïétique en empruntant à Varela et Maturana le concept d'autopoïèse et en développant les principes qui se cachent derrière la notion de couplage. Nous verrons en particulier comment ces couplages interviennent en redéfinissant des sous-environnements dans l'environnement de départ, appelé environnement englobant, dans lesquels, comme le souligne Varela et Maturana[29], entre la phénoménologie biologique et la phénoménologie physique, de nouvelles règles apparaissent conformément à une organisation émergente et locale. Nous verrons également comment une frontière apparaît avec le couplage, et comment elle possède une porosité indispensable au fonctionnement du couplage.
Avant de tenter une première description du phénomène de couplage autopoïétique, afin de préciser la notion de singularité, nous pouvons faire quelques observations dans des domaines suffisamment variés pour montrer l'étendue du dispositif. Nous parlerons ainsi de l'homme, sa constitution en tant qu'individu, mais aussi des lettres de l'alphabet latin, des nuages, des systèmes météorologiques dépressionnaires, des nuées d'oiseaux et des organisations humaines (association, état-nation, fédération,…).

II-C1) Les nuées d'oiseaux et autres contextes

Le fait que des individus se regroupent pour former un tout de niveau supérieur représente un certain nombre d'avantages dans les différentes stratégies de survie. Cela permet une exploration de l'environnement à un autre niveau que celui de chaque individu[30], mais aussi d'être plus robuste à cet environnement. Les travaux que nous avons menés ces dernières années sur le stockage de documents dans les réseaux informatiques ouverts, en s'appuyant sur la métaphore des nuées d'oiseaux[31], ont été directement liés à ces propriétés collectives de robustesse vis-à-vis de l'environnement. Il ne s'agissait pas uniquement de filer une métaphore, mais bien de recourir aux principes fondamentaux des systèmes vivants.

32 Ces règles naturelles ont été modélisées partiellement, mais relativement spectaculairement par C. Reynolds dans « *Flocks, herds and schools : A distributed behavioral model* », in Proceedings of the 14th annual conference on Computer graphics and interactive techniques, SIGGRAPH'87, p. 25-34, 1987.

33 Pour une étude complète de la classification des nuages, on pourra se référer au livre « *Le guide du chasseur de nuages* », Gavin Pretor-Pinney, aux éditions du Points Jean-Claude Lattès, collection Sciences, 2006. On y trouvera p. 21-22 le tableau de classification des nuages selon le système latin de Linné qui est basé sur l'altitude et l'aspect des nuages. Il ne parle pas des caractéristiques dynamiques dues à l'environnement comme les effets du vent sur leurs déplacements et bien sûr sur leur formation (évolution de leur forme)… Les formes des nuages sont tellement riches pour nourrir l'imaginaire que la peinture traditionnelle chinoise leur firent jouer un très grand rôle, à tel point que les théories picturales leur font une place importante, et disposent d'une terminologie extrêmement riche et précise pour distinguer tous les types de nuages et de phénomènes atmosphériques.

34 « *…Il peut paraître osé de rapprocher les phénomènes météorologiques de ceux du domaine organique, – aussi laisserons-nous la parole à un spécialiste de renom. Dans son ouvrage : "Die Dynamik des Zyklonen", P. Raethjen parle en ces termes des zones de basse pression (cyclones) : Le cyclone possède une biographie typique, avec des phases bien caractérisées d'apparition, d'évolution et de vieillissement. Il ne se propage pas à la manière d'une onde, en s'étendant dans l'espace, mais à la manière d'un être vivant : d'abord la naissance d'un jeune cyclone frontal qui sort du sein maternel (zone frontal) d'un cyclone central adulte… Étant donné que l'atmosphère se comporte comme un être vivant, nous ne la comprenons bien que lorsque nous la voyons et la traitons comme un tout…* », dans « *Das sensible Chaos/ Le Chaos sensible* », p. 107, Théodore Schwenk, dans la deuxième édition française, aux éditions du Centre Triades, 1982.

35 *CORBA* (« *Common Object Request Broker Architecture* ») est un standard international, discuté à l'OMG (« *Object Management Group* » – http://www.omg.org/spec/), apparu dans les années 1991, qui permettait d'interopérer les matériels et les systèmes informatiques à travers les réseaux. Dans ce standard et sous certaines conditions, l'objet *CORBA* n'existait que lorsqu'il était sollicité par l'environnement.

Revenons à l'articulation entre mobilité et autopoïèse. Avant de former une nuée, et également après en être sorti, chaque oiseau parcourt l'espace comme il l'entend, avec sa propre trajectoire, indépendamment de celles des autres oiseaux. Quant la nécessité l'exige les oiseaux se regroupent en vol pour former la fameuse nuée, au sein de laquelle chaque oiseau conserve une trajectoire propre, mais sous contraintes collectives. Ces contraintes apparaissent comme des règles naturelles[32] qui fédèrent le groupe en un individu de niveau supérieur, la nuée, possédant des caractéristiques propres (direction/ trajectoire, vitesse, densité, vélocité…). On voit ici que l'appartenance par couplage, à une entité de plus haut niveau, se superpose à une activité locale. Nous avons deux mobilités qui s'emboîtent. La situation est comparable à celle d'une personne qui se déplace à l'intérieur d'un avion en vol.

Les nuages suivent ce principe. L'observabilité d'un nuage caractérise son existence, donc la présence d'un (ou de plusieurs) couplage lui conférant des caractéristiques propres comme sa direction, sa vitesse de déplacement, son volume, éventuellement sa forme[33],… Mais si l'on observe de près le nuage, pendant qu'il se déplace, il se transforme et chaque élément constitutif évolue dans des directions indépendantes de la direction globale du nuage, et indépendamment des autres éléments constitutifs. Le nuage forme une espèce de nuée de particules d'eau. On va retrouver des principes analogues dans les systèmes météorologiques dépressionnaires avec l'organisation des vents dans une dépression[34]. La dépression, au sens entité de niveau supérieur, se déplace dans une direction donnée (trajectoire propre), linéaire ou non, avec une certaine vitesse. Au cœur de la dépression le vent se déplace de façon circulaire avec une vitesse variable, voire nulle en son centre, ou considérable en périphérie, indépendamment de la vitesse de l'ensemble.

On peut observer que la mobilité est souvent constitutive des couplages autopoïétiques, produisant des entités de niveau supérieur (n+1) avec leurs caractéristiques propres, y compris en terme de mobilité. D'une certaine façon la mobilité engendre la mobilité. Ce principe récursif permet à chaque niveau d'aborder l'exploration de l'environnement sous un angle nouveau, en passant par des sous-environnements. On peut se demander comment un individu peut se rendre compte qu'il appartient à un groupe/entité de niveau supérieur. Ce que l'on voit dans la nuée d'oiseaux, c'est que l'appartenance à une nuée n'annihile pas complètement le comportement individuel et autonome des oiseaux. Ils sont cependant contraints par la nuée. On retrouve ce principe dans les organisations sociales humaines. L'Europe contraint les règles des états membres, sans les remplacer toutes, qui elles-mêmes contraignent le comportement des individus…

Un nuage existe potentiellement, comme un objet *CORBA*[35], il suffit que les conditions nécessaires le fasse apparaître pour un observateur. La connaissance dans le système neuronal n'est perceptible et donc localisable uniquement pendant les stimuli, à travers des chemins d'interactions neuronales (plasticité). L'idée d'enregistrer une connaissance uniquement à partir des répercussions de son

36 On en connaît plusieurs comme la chasse ou encore le déplacement au long cours avec les vols en 'V'.

37 Cela pose d'ailleurs la question de l'intentionnalité du regroupement en nuée et plus généralement du couplage. Nous n'allons pas développer ici cette question, qui est très intéressante. Néanmoins nous pouvons ouvrir quelques pistes. L'intention avec un grand 'I', comme une direction, un objectif collectif à atteindre, ne peut être que la résultante ou l'action collective à mener pour finalement répondre à des attentes beaucoup plus individuelles. Quand les orques ou les loups fonctionnent en meute c'est pour que chacun récupère sa part. L'intérêt des processus autopoïétiques est de contraindre des individus dans une dynamique interrelationnelle, de telle sorte qu'émergent des comportements de groupe qui répondent à une intention préexistante (de façon consciente ou inconsciente comme la survie) ou co-construite (par la création de nouvelles intentions de façon opportuniste, comme par exemple la création d'une banque centrale européenne pour lancer une stratégie d'amortisseur de crise financière dans les banques étatiques et privées de l'UE) par ces interactions.

activation dans l'environnement où elle évolue, interdit toute observation statique de cette dernière.

Le couplage autopoïétique mobilise donc, sous contraintes d'un sous-environnement qu'il produit, des entités d'un niveau 'n' donné, afin de produire/faire émerger des entités d'un niveau supérieur (n+1), dont la durée de vie est directement assujettie à l'existence du couplage. Ce couplage n'altère en rien le fonctionnement des entités de niveau 'n'. Les oiseaux ne vivent pas dans une nuée en permanence. D'ailleurs suivant le type des oiseaux (canards, mouettes, moineaux, étourneaux,…) et la fonction recherchée[36] par le regroupement ponctuel (sous l'effet du couplage), on peut observer différentes typologies de regroupements/nuées[37]. De la même façon les poissons ne sont pas toujours en banc, les flocons de neige ne sont pas nécessairement associés à une avalanche, ni les particules d'eau dans l'atmosphère à un nuage. L'activité d'un cumulo-nimbus (le roi des nuages), sous sa forme active et donc la plus violente, dure à peine quelques heures. Ce couplage est donc un phénomène profondément réversible. La membrane qu'il génère (sous-environnement) est poreuse et peut donc être amenée à disparaître et réapparaître en divers lieux de l'environnement englobant.

Cela n'est pas en contradiction avec les lois irréversibles qui existent dans l'environnement englobant, comme le montre Ilya Prigogine dans son livre «*La fin des certitudes*».
Cette irréversibilité peut bien entendu avoir des répercussions sur les sous-environnements générés par les couplages. Une exposition trop forte aux radiations nucléaires peut provoquer des lésions irréversibles sur le système neuronal d'une personne et atteindre ainsi un couplage en cours.
Si l'on prend l'individu depuis sa naissance et jusqu'à sa mort, avec son enveloppe corporelle, il apparaît comme étant unique et faisant un tout. Son enveloppe corporelle ou membrane, ou encore sous-environnement, est en réactualisation permanente, comme le nuage dans le ciel. La correspondance est assez troublante. Le nuage existe devant nous avec une forme donnée (*perçue*), parfois interprétable. Si l'on ferme les yeux un instant, lorsqu'on les ouvrira à nouveau, en dehors d'un éventuel déplacement dans le ciel sous l'effet des vents, le nuage aura complètement changé de forme. Une observation continue du même nuage ne permet pas l'observation de ces changements de forme. Chaque constituant du nuage «*vit sa vie*» à l'intérieur de la membrane, permettant des transformations radicales de la forme globale. Ce changement de forme est beaucoup plus perceptible pour une nuée d'oiseaux.

En revanche l'évolution de chaque être humain suit le même principe que celui des nuages, avec un effet encore plus accentué d'invisibilité du changement continu pour un observateur donné (que ce soit un proche ou l'individu lui-même). Voir sa propre évolution, signifie fermer les yeux et regarder des photos du passé (discrétisation)… À la fin de sa vie sa membrane externe, avec les membranes

38 En hommage à l'interprétation par Philippe Léotard de la chanson « *La mémoire et la mer* » de Léo Ferré.

39 À l'image de l'apparition des unités autopoïétiques aux origines de la vie, décrites par Maturana et Varela.

40 C'est le cas de la formalisation par Reynolds des règles comportementales des oiseaux dans les nuées, pour ne citer que cet exemple.

internes des sous-environnements qui le composent, disparaîtront. Un observateur verra ainsi disparaître et réapparaître des humains, se ressemblant et pourtant différents, exactement comme l'on voit apparaître et disparaître des nuages dans le ciel, avec l'impression de les avoir déjà vu cent fois, mille fois…

II-C2) Singularité paradoxale versus singularité naturelle

Chaque individu est singulier, en quoi participe-t-il d'une singularité ? Nous devons maintenant préciser la mathématique bleue[38] de la singularité. Être unique ne signifie pas nécessairement qu'il existe une singularité associée. L'unicité est le point de départ d'une singularité potentielle, en devenir. Cette unicité originelle doit être reprise, fertilisée, transformée par d'autres entités dans un contexte de partage de contraintes exogènes, autrement dit dans un environnement. Ceci donnera naissance à de nombreuses autres unicités dérivées, c.-à-d. identiques mais différentes, simultanément et en des points différents de l'environnement[39]. C'est dans cette multiplicité que pourra apparaître une singularité.
Une singularité est constituée d'une forme canonique nécessaire aux abstractions de niveaux supérieurs (sa face visible, tangible), mais aussi d'un ensemble de règles/lois consubstantielles aux couplages autopoïétiques sous jacents.
Les environnements de ces couplages autorisent/permettent/rendent possibles/instancient ces règles informelles, potentiellement formalisables dans certaines situations, au moins de façon approximative[40]. Produire des singularités revient à définir/provoquer l'établissement de règles/lois qui permettent les couplages. C'est la rémanence de ces règles qui font émerger les singularités «visibles» au sens utilisables dans des niveaux [d'abstraction/représentation] supérieurs.
Chaque entité du système étudié ne peut contribuer qu'à une singularité à la fois dans un point de vue donné, rendant nulle l'intersection entre deux singularités distinctes dans ce même point de vue. Nous verrons avec l'exemple de l'alphabet latin, comment chaque lettre est une singularité et qu'une occurrence d'une lettre donnée ne peut définir/appartenir qu'à une seule singularité, sa lettre canonique, à l'exception des lettres mal formées, considérées comme incertaines, inintelligibles (des non-formes, éventuellement en devenir). Nous verrons également, dans le domaine du langage naturel, qu'une lettre donnée (sa forme canonique) peut appartenir simultanément à des singularités distinctes, qui évoluent nécessairement dans des points de vue distincts. On entend par point de vue un ensemble de règles/contraintes dans un sous-environnement particulier.

Toute entité participant à la définition d'une singularité peut en sortir, suivant sa propre évolution, et celles des règles de la singularité associée. Dans le système de classification des espèces animales vertébrées, une espèce peut être rangée provisoirement, par défaut, dans une

41 René Thom (mathématicien français à l'origine de la théorie des catastrophes) répondant à Jacques Lacan (philosophe français) qui lui demandait : « d'après vous quelle est la limite du vrai ? », de répondre : « ça n'est pas le faux mais l'insignifiant ! ».

catégorie donnée, puis en changer à la suite d'apparitions d'autres espèces suffisamment proches entre elles et éloignées des espèces existantes, pour définir une nouvelle catégorie.
Toute singularité définit son sous-environnement propre dans lequel une mesure de différenciation peut être faite entre chaque entité constituante et la forme canonique associée. Cette forme canonique évolue sans cesse, par le jeu des entrées-sorties de la membrane, d'où sa porosité. Une singularité peut être soit isolée, on la nommera dans ce cas une *singularité paradoxale*. En effet cela signifie que cette singularité ne sera pas reprise dans un niveau supérieur (n+1) à des fins compositionnelles.

À l'inverse et plus généralement, une singularité pourra être composée avec d'autres singularités dans un niveau supérieur. C'est le cas des lettres de l'alphabet, afin de produire les mots. On parlera alors de *singularité plurielle* ou *singularité naturelle*, autorisant les niveaux d'abstraction. Dans ce dernier cas, on peut définir l'espace de ces singularités plurielles, dans lequel certaines d'entre elles seront proches les unes des autres, alors que d'autres seront éloignées. Nous verrons avec l'exemple de l'alphabet que la singularité plurielle de la lettre 'X' est très proche de celle de la lettre 'V' ou de la lettre 'Y', qui sont un peu plus éloignées de celle de la lettre 'A' dans l'espace associé. Cette notion de distance reposera sur le concept d'incertitude ou d'insignifiance[41]. Nous verrons que les règles associées aux singularités de 'X' et de 'V' permettent le passage entre 'X' et 'V' de façon très «naturelle». Ceci signifie que lorsque deux singularités sont proches dans l'espace associé, il existe une zone d'incertitude entre elles conduisant à confondre deux occurrences de ces singularités ou tout du moins ne pas pouvoir décider si telle occurrence définit l'une ou l'autre des singularités adjacentes. Cela décrit une sorte de continuum dans l'espace associé entre ces singularités.

L'alphabet latin est donc un ensemble de vingt-six singularités naturelles, une par lettre. Nous verrons comment un ensemble restreint de traits peut construire ces lettres. Chacune d'elles se caractérise par l'ensemble des occurrences passées, présentes et à venir de cette lettre. Un autre ensemble de traits aurait pu être choisi, de la même façon, avec cet ensemble de traits, d'autres lettres auraient pu émerger.

Des couplages longs et évolutifs ont permis le choix parmi les possibles, pour fixer ces vingt-six singularités. Si nous examinons maintenant les nuées d'oiseaux, la nuée n'est ni une singularité paradoxale, ni même une singularité naturelle. La nuée ne représente aucun de ses membres constituants, les oiseaux. Il émerge bien une entité d'un niveau supérieur, avec ses propres propriétés. C'est aussi le cas des bancs de poissons ou des regroupements d'orques pour la chasse aux lions de mer. Il nous faut revenir au concept d'émergence, pour affiner les choses.

42 L'actualité, fin 2014 – début 2015, de l'Union Européenne (UE) parlait (les allemands surtout, Mme Merkel en particulier) à l'époque du retrait ou non de la Grèce de l'UE en fonction du résultat des élections à venir en Grèce. Dans un passé relativement proche de cette période, c'était l'existence même de l'UE qui était en cause. Nonobstant la survivance du mille-feuilles territorial français, lui aussi sur la sellette, l'UE n'est juste qu'au niveau n+1 des états membres. Cela donnerait à penser que l'empilement des niveaux reposant sur un empilement des couplages en équilibre, apportant ainsi la cohérence à l'ensemble, possède une probabilité décroissante avec l'altitude ou le niveau d'emboîtement atteint.

II-C3) Émergence constitutive versus émergence compositionnelle

Des processus émergents sous-tendent l'apparition de singularités. Il en existe au moins de deux types: l'*émergence constitutive* ou encore appelée *émergence homogène*, et l'*émergence compositionnelle*, appelée *émergence hétérogène* par opposition/complémentarité à la première forme d'émergence. L'apparition des vingt-six lettres de l'alphabet relève de l'émergence constitutive, alors que les nuées d'oiseaux procèdent d'une émergence compositionnelle. Il est intéressant de regarder l'articulation de ces différents mécanismes dans l'apparition des singularités.

L'émergence constitutive permet l'existence de singularités (les occurrences de la lettre 'A' définissent la lettre 'A'), chacune d'elles représentée entre autres choses par une forme canonique (partie visible et émergente du processus). Cette singularité peut être utilisée dans les niveaux supérieurs avec d'autres singularités dans un processus d'émergence compositionnelle pour produire une nouvelle unicité (par exemple les mots). Cette dernière peut, avec d'autres unicités de nature voisine, produire une nouvelle singularité, et ainsi de suite… Cela fait apparaître une chaîne, presque sans fin de successions d'émergences constitutionnelles et d'émergences constitutives, produisant des singularité dans des niveaux d'abstraction (sous-environnement) de plus en plus élevés/ emboîtés. Dans le système des connaissances humaines et plus particulièrement celui du langage naturel, les différents niveaux d'emboîtement peuvent être nombreux, allant par exemple jusqu'à la notion de style d'un auteur. En revanche dans les organisations sociales humaines, le niveau d'empilement est beaucoup moins profond, car chaque niveau de singularité nécessite que les couplages, qui le produisent, fonctionnent toujours[42]. Il en va ainsi de l'équilibre de l'ensemble. Dans le corps humain si le couplage interne d'un organe vital avec son environnement ne fonctionne plus, c'est l'ensemble de l'édifice qui s'écroule. Cela peut, dans certains cas, être une très bonne chose. Nous voulons parler par exemple des super-tsunamis, comme celui de Fukushima ou celui plus ancien de l'océan Indien, ou géographiquement plus proche de nous, la tempête de Noël 1999. Dans ces situations extrêmes, un niveau supplémentaire et extrêmement inusuel de couplage est apparu. En 1999 les côtes françaises subirent l'effet composé (couplage) de deux dépressions dans un niveau supérieur à celui des simples tempêtes; de la même façon le super tsunami conjugua le double effet de plusieurs vagues pour produire une vague géante (vingt mètres de haut). Nous ne sommes pas trop demandeur de la réalisation de ce genre de couplage. Heureusement leur nature relativement improbable nous rassure.

Dans le cadre des lettres de l'alphabet latin, l'émergence compositionnelle se situe au niveau des traits qui les composent et de leur agencement dans le plan, voire, le cas échéant, dans l'espace tridimensionnel. Ensuite, un couplage autopoïétique permet la définition d'une forme canonique associée qui est construite dans l'environnement, à partir de toutes les occurrences de la lettre, et prend

son sens dans le sous-environnement du langage écrit. Sur la feuille de papier, un ensemble de traits ne relevant d'aucune singularité alphabétique, n'apparait donc pas dans le sous-environnement du langage écrit; le couplage n'opérant pas, l'interprétation de ces traits reste dans l'espace englobant comme une trace non signifiante. Dans le cas où le couplage réussit, l'occurrence identifiée comme une instance d'une singularité-lettre peut être composée avec d'autres instances présentées de façon concomitante, pour identifier dans un niveau supérieur une autre singularité, telle qu'une syllabe, un mot, un bigramme un morphème… (cf. la suite de l'article).

II-C4) Retour aux domaines sensibles, naturels et artefactuels

Il existe des liens étroits entre ces phénomènes de couplage et l'art. Le peintre est un être errant, un voyageur de l'immobile. Son approche consiste à faire des «pas de côté» dans son espace des possibles. Un couplage vertical (émergence constitutionnelle) de 1er niveau le conduit à produire des tableaux (au sens générique d'une production artistique, une œuvre).
En faisant un tableau il entre en interaction avec de nombreux éléments (matériels, formels, son propre regard, son troisième œil, le regard des autres, l'expérience, les savoir-faire,…) qui le conduisent à rechercher le couplage qui amènera le tableau dans sa version finale, donc satisfaisante. Le couplage ne marche pas à chaque tentative. Quand le couplage opère, le tableau prend forme. Survient alors une période pendant laquelle les choses peuvent être améliorées aux yeux du peintre (à l'image de la nuée qui se déforme localement sur elle-même en faisant des replis spectaculaires).
Il se peut même que de couplages en couplages locaux (sur une partie de la toile) le tableau se transforme en un autre tout aussi intéressant que le premier, aux yeux du peintre. La toile, une fois terminée, est le résultat du ou de ces couplages locaux successifs.
C'est la trace visible qui reste de ces processus, comme pour la partition de musique, le roman ou la démonstration mathématique. Ces traces mettent en jeu l'échange et le partage dans d'autres processus de couplage avec le spectateur, sur lesquels nous ne nous étendrons pas ici. L'artiste ne fait que très rarement un seul tableau. Il apprend, progresse et surtout explore en travaillant une œuvre.

Les plus grands peintres ont plusieurs périodes bien distinctes au cours de leur carrière (parmi quelques-uns des plus marquants on peut citer Matisse, Picasso ou Braque). Ces périodes jalonnent la production des peintres, de grandes ruptures/bifurcations. En fait le couplage vertical produisant un tableau intervient dans une période donnée de l'œuvre du peintre. Ce sont autant d'occurrences qui cristallisent davantage la période en cours, jusqu'à la rupture. Nous sommes dans un couplage d'un autre niveau, provoquant une émergence constitutive. Les occurrences (ici

43 «*…Une vague, en général, n'engendre pas de courant. L'eau qu'elle traverse décrit seulement des cercles sur place… Quand la vague meurt sur un rivage plat, les mouvements circulaires deviennent elliptiques et se résolvent finalement en va-et-vient horizontal. Le mouvement rythmé des vagues engendre une série de courants dont le sens alterne incessamment. Mais un courant peut aussi se former sur le "dos" d'une vague, quand le vent souffle assez fort pour que la surface de cette vague se mette à "couler" sur le reste de son épaisseur. L'eau qui coule ainsi sur le dos de la vague se meut alors plus vite que la vague elle-même, franchit sa crête et se jette dans sa "vallée". Une des couches d'eau se glisse sur l'autre et la dépasse. Il en résulte des enroulements et d'autres mouvements divers…*» Extrait du livre de Théodore Schwenk, «*Das sensible Chaos/Le Chaos sensible*», p.29 dans la deuxième édition française, aux éditions du Centre Triades, 1982.

44 Ce paragraphe est issu du livre «*Quand la Beauté nous sauve. Comment un paysage ou une œuvre d'art peuvent changer notre vie*», de Charles Pépin, aux éditions Poche Marabout, 2014.

les toiles) font émerger la singularité (ici la période). L'espace des possibles d'un peintre est plus ou moins étendu. Son errance est la chose la plus complexe qu'il doit apprendre à dominer ou à subir. Il ne faut ni faire du sur place, ni sortir trop vite d'une plaine fertile.

La mer peut aussi être vue sous l'angle des couplages. Lorsque la marée descend un mouvement lent, continu et immuable fait reculer la mer sur la grève, quelles que soit les conditions atmosphériques (vents, température,…). Si dans le même temps un fort vent souffle du large en direction de la côte, il se forme en surface un système de vagues qui arrivent sur la côte. Nous avons un double système qui semble s'opposer et pourtant se combine.
Des vagues apparaissent par le couplage de l'effet gravitationnel des marées, alors que d'autres vagues apparaissent sous l'effet du vent. L'environnement de ces phénomènes et leur couplage associé possèdent leurs propres règles[43].

Le code de la route est également un bon exemple pour les couplages autopoïétiques. Le sous-environnement du code de la route est l'espace où les voitures circulent.
Il y a bien porosité entre ce sous-environnement et l'environnement englobant, rien n'empêche les voitures d'aller sur les trottoirs. Dans ce sous-environnement, des règles ont été édictées afin de garantir aux automobilistes, mais pas seulement (aux piétons,…) des déplacements en toute sécurité, ou presque…
Les trottoirs, en ville, forment un sous-environnement, propre aux piétons. Il possède lui aussi des règles comportementales, mais nettement moins élaborées/ formalisées que pour celles relatives aux voitures. Nous sommes capables de nous croiser sur un trottoir sans trop risquer la collision, même si nous l'avons tous vécu un jour ou l'autre.

Nous ne pouvons pas ne pas parler du métier d'enseignant et d'une de ses activités rébarbatives qui est la correction de copies. On peut décrire cette activité en évoquant la notion de couplage autopoïétique et d'apparition de singularité. Par expérience les dix ou vingt premières copies d'une série sont laborieuses, voire pénibles à corriger. Les suivantes le sont nettement moins. Que se passe-t-il? Il y a principalement deux phénomènes concomitants: la mise en mémoire par le correcteur du barème et les réponses aux questions. Les réponses aux questions peuvent être assez surprenantes, même si le correcteur a fait de véritables efforts pour les envisager toutes.
L'ambiguïté résiduelle dans la formulation des questions, couplée à l'imagination et les connaissances des étudiants peuvent produire, dans certains cas, des réponses aussi inattendues que pertinentes.
Dans la mesure où le correcteur a la hantise de l'inéquité dans sa notation, il lui faut corriger un certain nombre de copies pour ajuster son barème. Il s'agit bien d'une émergence constitutive, chaque nouvelle instance (copie) améliore/co-construit la forme canonique en devenir (notation effective en opposition au barème a priori) dans le contexte opérationnel (les étudiants).

Une réponse particulière d'un étudiant sur une question donnée, peut même conduire le correcteur à reprendre toutes les copies déjà corrigées sur la base d'un nouveau barème.

Le langage conduit lui aussi à des couplages en se basant sur des règles permettant la génération d'entités de niveau supérieur. Une bonne maîtrise de ces règles suffit pour produire un texte correct, mais pas nécessairement pour produire un bon texte littéraire. Nous touchons ici toute la problématique du rapport entre la stylistique et la traduction d'œuvre littéraire.

Dans le domaine des arts, certains artistes jouent avec ces règles pour les déconstruire; ils produisent des couplages inédits et pourtant intelligibles. C'est le cas du free jazz.

En littérature on trouve des exemples célèbres comme l'Oulipo, avec l'écriture automatique de Georges Perec, mais aussi des exemples moins spectaculaires et pourtant tout aussi efficaces comme «*Voyage au bout de la nuit*», de Louis Ferdinand Céline, où l'auteur utilise des règles du langage parlé qu'il transpose dans le langage écrit.

Pour finir évoquons rapidement en philosophie[44] les notions de *jugement réfléchissant* et de *jugement déterminant*. Ces concepts sont issus de l'esthétique kantienne développée dans «*Critique de la faculté de juger*» Le jugement réfléchissant va du cas particulier vers un jugement général, voire universel sans avoir recours à aucun critère précis. C'est le cas d'un paysage dont on énonce un jugement partagé, comme «c'est beau». Malgré l'absence totale de critères de jugement nous n'avons aucun doute sur notre propre jugement.
Alors que le jugement déterminant va du cas général vers le cas particulier. On dispose d'une catégorie générale/ canonique et d'un critère qui préexiste, et on examine le cas particulier pour dire «c'est bien», «c'est faux»,… Dans ce cas, bien qu'il existe des critères pour porter un jugement nous doutons de notre jugement.

Nous voyons à travers ces concepts qu'il existe des relations subtiles entre les cas particuliers et les formes canoniques, y compris dans le domaine des idées.

Une expérience de couplage autopoïétique

45 D'après Eric Havelock, dans Havelock, Eric A., 1981, «*Aux origines de la civilisation écrite en Occident*», traduit de l'anglais par E. Escobar Moreno, Paris, Maspero, cent-cinq pages, les trois conditions théoriques suivantes doivent être satisfaites simultanément pour pouvoir parler d'alphabet : «*il doit être rendu compte de tous les phonèmes de la langue, sans exception ; le nombre de caractères ne doit pas dépasser un chiffre situé entre vingt et trente ; les caractères ne doivent pas avoir un double ou triple emploi, leurs équivalences phoniques doivent être bien définies et invariables.*»

46 Dans «j'ai mal», si l'on descend 'mal' à 'ma' il y a perte de sens, donc 'mal' est un morphème. «redemanderons» comprend quatre morphèmes ('re'+'demand'+futur+1re personne du pluriel). «au fur et à mesure» ne comprend qu'un seul morphème, il s'agit d'une structure figée.

47 Les morphèmes lexicaux représentent une liste ouverte (adjectifs, noms,…), alors que les morphèmes grammaticaux représentent une liste fermée (désinences des conjugaisons verbales, prépositions,…) «Les boxeurs souffrent» est décomposé en sept morphèmes : dont deux lexicaux ('box' et 'souffr') et quatre grammaticaux ('le', 's', 'eur' et 'ent').

48 «pâle» et «pile» n'ont pas le même sens et pourtant le phonème /i/ n'a pas de sens, c'est une unité distinctive. Le morphème 'nous' peut être décomposé en deux phonèmes [n u], en les changeant le morphème change de sens.

49 «Baraque» possède le phonème /R/, qu'il soit prononcé avec un [R] uvulaire ou un [r] roulé ce sont deux réalisations du même phonème /R/.

50 Georges Mounin, dans «L'écriture script, Introduction à la sémiologie», aux éditions de Minuit, 1970, p.137-144, a décomposé en formes graphiques élémentaires les lettres de l'écriture latine pour les majuscules écrites en script. L'alphabet latin met en œuvre un système de douze traits graphiques minimaux, dont huit sont droits, et quatre arrondis. Il constate également que sur vingt-trois lettres, dix sont formées de trois traits graphiques, huit de deux traits, trois d'un seul trait, et deux de quatre traits. 'A'='/'+'-'+'\' ; 'B'='|'+'⊃'+'⊃',… la haste verticale '|' est utilisée treize fois ('B', 'D', 'E', 'F', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'R', 'T'), le trait horizontal '-' sept fois ('A', 'E', 'F', 'H', 'L', 'T', 'Z'),… «*Tous ces traits graphiques minimaux permettent de faire un ensemble d'oppositions distinctives, qui, dans leur grande majorité, sont des oppositions que l'on qualifiera de privatives, parce que l'un de ces membres se caractérise par l'absence de l'élément graphique qui est propre à l'autre. La plupart des lettres à deux traits graphiques s'opposent en effet à des lettres à un seul trait par la présence d'un trait supplémentaire…*», dans «*Les lettres du latin : description sémiologique, fonctionnelle et graphématique*», p5, Christian Touratier, Université de Provence. http://www.paris-sorbonne.fr/IMG/pdf/Semiologie_fonctionnement__graphematique-2.pdf.

51 Les groupes de lettres sont associés à un phonème unique. On trouve par exemple 'ss' pour les bigrammes ou 'œu', 'oin', 'ain' ou encore 'ein' pour les trigrammes.

Si l'on aborde la question du langage naturel et plus particulièrement celle des processus liés à la reconnaissance et à l'engrammage du langage naturel, cela pose le problème de la représentation des connaissances en machine. C'est un très bon exemple pour illustrer le paradoxe de la singularité, dans la mesure où il s'y prête et où l'on peut conduire un début d'expérimentation.

L'étude du langage naturel (parlé et écrit étant liés) a conduit à de très nombreux travaux, dans des directions parfois différentes, qui peuvent même s'opposer sur certains aspects.
Au départ il y a le signe graphique, les alphabets[45].
Ensuite il y a plusieurs aspects qui se complètent, s'interpénètrent, s'articulent.
Les grandes lignes de l'articulation d'une langue, qui vont nous permettre d'illustrer les mécanismes d'émergences et de singularité, sont les suivants :

- la *morphologie* regroupe l'étude des formes attachées au sens, et plus particulièrement fait la distinction entre le concept de mot, trop vague, et celui de morphème, beaucoup plus opérationnel[46] ; on trouve d'ailleurs des morphèmes lexicaux et des morphèmes grammaticaux[47] ;
- la *phonétique* étudie ce qui est émis et donc perceptible : les «phones» ;
- la *phonologie* étudie comment la langue regroupe les phones en catégories appelées des phonèmes.

II-D1) L'alphabet latin

Dans ce contexte les morphèmes représentent l'unité minimale de sens que l'on ne peut diviser en unités plus petites. Au palier inférieur de la chaîne, on passe ensuite au niveau phonologique. Les phonèmes, basés sur les phones, sont totalement dépourvus de sens. Ils ont une fonction distinctive dans les morphèmes qui les intègrent[48]. Ils peuvent permettre le classement des sons, de la chaîne parlé selon une série de traits pertinents[49]. Les phonèmes, vus comme des unités distinctives minimales, produisent des différences négatives qui créent du sens. On parle de la double articulation du langage.

En partant du plus général (intégralité du discours d'un homme politique, comparaison de plusieurs œuvres d'un même écrivain,…) on essaie d'aller vers l'unité minimale de sens. On peut analyser les différentes parties du discours, puis chaque partie en sous-parties, puis en phrases. On arrive à la première articulation : le morphème. La deuxième articulation concerne l'unité distinctive, le phonème, qui permet justement la différenciation entre plusieurs morphèmes.
En plus de l'encodage oral des morphèmes et/ou des mots, un certain nombre de langues ont ajouté un encodage graphique qui peut aller jusqu'à la production des alphabets[50]. Or il existe un lien entre les signes et le sens, représenté par les morphèmes, qui passe justement par les phonèmes. Il s'agit des graphèmes. Les lettres, ou les groupes de lettres (bigrammes et trigrammes[51]) sont des réalisations

52 «…*On parle de graphème, en entendant par là une unité graphique minimale : "le graphème peut être défini comme la plus petite unité (lettre ou groupe de lettres) de la chaîne écrite ayant une référence phonique et/ou sémique dans la langue parlée" dans Catach, 1980, "L'orthographe française", Paris, éditions Nathan, p. 334… pour travailler en graphématique comme on le fait en phonologie, il faudrait définir le graphème comme une unité graphique fonctionnelle, que l'on noterait entre deux chevrons, et considérer que les lettres ne sont que des allographes ou des réalisations graphiques de graphèmes, tout comme les sons ne sont que des allophones ou des réalisations phoniques des phonèmes. Si donc la lettre 's' correspond à un phonème /s/, quand rien dans le contexte ne la conditionne, on a affaire à un graphème que l'on notera <s>, celui-ci faisant correspondre la lettre 's' au phonème /s/. Mais quand, dans un contexte particulier précis, cette même lettre correspond à un phonème /z/, elle représente un autre graphème, que l'on notera <z>. Elle n'est pas une variante du graphème <s>… mais une variante du graphème <z>. Et inversement, on dira que les deux lettres 'ce' sont des allographes du graphème <s>, puisqu'elles correspondent au phonème /s/. On arrivera ainsi à la conclusion qu'en français, le graphème <s> est représenté par plusieurs allographes, à savoir la lettre 's', le digramme 'ss' entre voyelles, la lettre 'c' devant un 'i' ou un 'e', ou le digramme 'ce',… suivant les contextes : Le graphème devenant ainsi véritablement une unité fonctionnelle comparable au phonème, et la lettre, une unité substantielle, comparable au son… La lettre 's' seulement est l'allographe de base qui donne son nom au graphème <s>, parce qu'il n'est entraîné par aucun contexte, tandis que 'c', 'ce', 'ss' ou 't' sont des allographes de ce même graphème <s> qui sont conditionnés par leur contexte graphique, devant 'e' ou 'i', pour la lettre 'c', entre deux voyelles pour les deux lettres ou digramme 'ss', et devant 'ion' pour la lettre 't',…»* dans Christian Touratier, Université de Provence. http://www.paris-sorbonne.fr/IMG/pdf/Semiologie_fonctionnement__graphematique-2.pdf.

graphiques contextualisées des graphèmes. Selon cette acception, un graphème sera alors vu comme une entité abstraite[52]. Cette entité abstraite s'instancie par des allographes, qui ne sont que des représentations écrites concrètes d'un graphème en fonction du contexte (les lettres dans le voisinage immédiat). Par exemple le graphème noté <S>, qui représente le phonème /S/, possède quatre instanciations contextualisées: allographe$_1$ (la lettre 's' seule, sans voisinage), allographe$_2$ (la lettre 't' suivie par exemple de 'ion'), allographe$_3$ (le bigramme 'ss' entre deux voyelles) et allographe$_4$ (la lettre 'c' suivie de 'e' ou de 'i'). Autrement dit un graphème est une singularité, obtenue par émergence constitutive des différents allographes (instances) qui le définissent dans un environnement graphique donné (positionnement des lettres dans les morphèmes).
L'engrammage d'un texte passe par un processus de reconnaissance des caractères et réciproquement. Cette étape peut se heurter à des ambiguïtés si l'on travaille sur l'écriture manuscrite. Cela peut être le cas pour le 'a' et le 'd'. Si l'on se base sur la décomposition de Georges Mounin (cf. la Note 50) on voit que les lettres 'Y', 'V' et 'X' utilisent les mêmes traits graphiques ('\' et '/'), mais avec des positionnements spatiaux et des longueurs légèrement différents. Dans le cas d'une écriture manuscrite ces positionnements peuvent jouer un rôle très important dans la reconnaissance.

En fait pour décrire une lettre on s'aperçoit que l'on peut utiliser des règles de type oppositions distinctives, mais cela ne suffit pas. On pourrait recourir à des règles de positionnement dans l'espace des traits relativement les uns aux autres. Le problème étant de reconnaître une lettre parmi les autres «proches»; comme 'V' pourrait être considérée comme très proche de 'X', si l'on se réfère aux modifications spatiales à faire pour passer d'une lettre à l'autre. Nous introduisons chaque lettre comme une forme canonique de l'alphabet. Pour savoir si une occurrence (variante écrite) d'un 'V' correspond bien au 'V' canonique et non au 'X' canonique, cela revient à appliquer à l'occurrence présentée les règles qui définissent la forme canonique 'V'.

Dans notre cas précis on pourrait énoncer les règles suivantes: si les traits '/' et '\' se coupent à leur extrémité inférieure, alors nous avons un 'V'; s'ils se coupent en leur centre, alors nous avons un 'X'.

Chaque lettre définit un attracteur dans l'environnement d'assemblage des traits. La forme du bassin autour de chaque attracteur caractérise la zone d'incertitude qui entoure la reconnaissance de chaque lettre. Les bassins peuvent être plus ou moins adjacents les uns aux autres (par exemple 'X' et 'V'). Concrètement cela signifie que l'on peut passer d'une lettre à une autre par simple transformation continue (par simple glissement du '\' le long du '/' dans le cas de 'X' et de 'V').

Mais l'existence d'un très grand nombre d'occurrences possibles, quasiment infinie, peut nous amener à ajouter

53 Cette question sera d'ailleurs posée très vite en philosophie comme le rapporte Michel Serres dans son livre «*L'Incandescent*», p.150-151 en mentionnant le concept de «géométral» qui fait écho à la notion de forme canonique présentée ici : «*…Parfaitement au fait de ces particularités respectives liées à des vues irréductibles, marques, disaient-ils, de notre limitation de créatures finies, les philosophes classiques, Leibniz par exemple, se demandaient s'ils pouvaient concevoir et si existait un point de vue privilégié, une sorte de lieu sommant tous les sites possibles, à partir duquel celui qui s'y trouverait verrait l'objet tel quel ou en soi. Nous ne le voyons jamais, nous croyons cependant qu'il existe, pusique nous disons : le vase, dans l'exemple de la perception ; ou la bâtisse pour les dessins d'architecte, comme nous disons langue ou musique, malgré ou en raison des variations d'accent, de style, d'instrument ou de partition. Leibniz concevait là l'intégrale des points de vue ou des scénographies humaines indéfiniment déployées. Il repoussait ce site à l'infini, comme si le cône de vision propre à chacun devenait, à la limite, un cylindre. Situé ainsi hors du fini, Dieu voit l'intégrale des profils, révélant l'objet dans sa réalité. Cette intégrale, Leibniz la nomme le géométral de l'objet ou son ichnographie. Lorsque nous parlons de la langue, alors que nous ne lisons que des styles et que nous n'entendons que des mots de métier dits en des accents obliques…, nous désignons en silence ou aveuglément ce géométral. Nous pensons au quotidien l'universel sans le savoir. Leibniz ajoutait même volontiers que nous savons lire le géométral derrière chaque profil…*»

d'autres règles pour mieux prendre en compte toutes ces situations.
Par exemple on pourrait énoncer pour 'V' la règle suivante : l'extrémité du trait descendant doit correspondre à l'origine du trait montant. Pour 'X' on pourrait dire : les deux traits constitutifs doivent avoir un point de contact qui n'est pas une extrémité ni pour l'un, ni pour l'autre.

La lettre 'X' n'est pas loin non plus de la lettre 'A', ni de la lettre 'V', elles-mêmes proches l'une de l'autre.
Pour différencier 'V' et 'A' de 'X' on peut dire que les traits constitutifs ne se croisent ou ne se rencontrent qu'à une extrémité du segment, sinon, en partant des mêmes traits, on pourrait passer de 'X' à 'V' par simple glissement de ces derniers.

Encore pour 'X' nous pouvons dire «deux barres obliques, de même longueur, avec un angle de 90° entre elles et qui se coupent par le milieu». Ceci est une règle générale qui décrirait le 'X' canonique/le modèle que l'on apprend à l'école primaire.
Mais on n'écrit jamais un 'X' idéal, même avec les ordinateurs, car chaque police apporte ses variations. On pourrait donc ajouter d'autres règles pour apporter de la souplesse. Par exemple l'angle des traits '/' et '\' entre eux peut varier, le rapport de longueur entre ces deux traits peut être de 1 à ⅛, le croisement de ces deux traits peut être compris entre la moitié et le ¼ de chaque trait… On peut même ajouter des règles de dépendance entre les traits, comme : si l'un des traits est plus petit que l'autre, le croisement devra plutôt se faire à tel endroit avec un angle entre tel et tel degré.

La réalité d'une lettre est dans les écarts possibles à sa forme canonique, disons académique, celle de la représentation conceptuelle/désincarnée/modélisée[53].

Pour engrammer chaque lettre canonique de l'alphabet, nous allons présenter toutes les occurrences possibles (un certain nombre) d'une lettre donnée et reconnue comme telle. Chaque lettre aura ainsi un engramme correspondant à son attracteur. Une forme canonique permet d'identifier tel ou tel graphisme (façon d'écrire) singulier comme comportant au moins un trait pertinent qui permet de l'assimiler (le considérer comme semblable) au modèle.

Nous retrouvons toujours cette question de la distance à la forme canonique émergente. L'encodage/engrammage, quel qu'il soit (quelle que soit la singularité en jeu) doit inclure implicitement les règles de mesure de cette distance à la forme canonique de niveau supérieur. Jean-Claude Ameisen parle de différenciation cellulaire.
Cela peut être plus ou moins intuitif, comme par exemple pour la constitution graphique des tracés, encore que, rendre cette mesure opérationnelle n'est pas aussi simple. C'est moins évident dans le cas des graphèmes qui se construisent à partir de leurs allographes, sur la base du phonème associé.

54 « *La sculpture du vivant. Le suicide cellulaire ou la mort créatrice* », Jean-Claude Ameisen, Éditions du Seuil, collection Points/Sciences, 2003.

II-D2) Complexité artificielle

En détaillant la construction/modélisation des signes de l'alphabet latin, ainsi que l'articulation entre ces signes et le sens porté par la langue naturelle (morphologie), nous avons voulu montrer concrètement sur des exemples «simples» comment les articulations et la complexité apparaissent, illustrant la notion de singularité telle que nous l'abordons dans ce texte. En effet la complexité est un enchaînement de couplages autopoïétiques aussi riches que divers et sans cesse en mouvement.

D'autre part nous avons commencé des simulations sur la représentation des signes de l'alphabet par des processus de couplage autopoïétique. Ceci devrait nous permettre à terme de mieux comprendre ces mécanismes et, pourquoi pas, de construire de la complexité artificielle.

Sans entrer ici dans le détail des heuristiques que nous utilisons, nous nous basons sur une transposition des trois grands phénomènes complémentaires[54] en jeu dans la métamorphose d'une cellule-œuf en embryon, à savoir:

- la division ou le dédoublement cellulaires (produisant la multitude): division de l'espace topologique de représentation de la forme canonique (lettre/signe) en sous-régions faisant apparaître de nouvelles entités d'encodage;
- la différenciation cellulaire (création de l'asymétrie, la diversité, la régionalisation et la complémentarité): mesure d'une distance à la forme canonique par un système dynamique de stimuli/réponse de l'organisation topologique de cette forme aux nouvelles instances présentées;
- la migration (elle répartit et recompose dans l'espace cette diversité): l'encodage est réalisé via la migration de vecteurs de stimulations dans la topologie, les éléments d'engrammage se déplaçant eux-mêmes dans l'environnement, assurant ainsi la plasticité de la forme canonique.

Enfin nous avons également un quatrième phénomène, toujours décrit par Jean-Claude Ameisen, qui est celui de la mort cellulaire.
En effet l'auto-organisation, qui pilote l'encodage de la forme canonique, explore l'espace d'états (partitionnements possibles de son environnement), afin d'obtenir une représentation doublement optimale:

- quantitativement, en terme de coût (mobilisation de ressources);
- mais aussi qualitativement, en terme de représentation la plus fine possible de la singularité correspondante.

Cette représentation doit être capable de reconnaître ses instances, c.-à-d. celles qui ont participé à son émergence, mais aussi celles qui pourront venir dans le futur.
Elle devra en outre représenter toute la finesse des zones d'incertitudes entre les différences singularités, voisines dans l'espace correspondant de niveau supérieur.

55 « *Les codes d'effacement ont été proposés pour le stockage décentralisé dans le but de diminuer le coût de stockage engendré par la réplication au détriment du degré de réparation…* » p. 50 dans la thèse de Benoît ROMITO, « *Stockage décentralisé adaptatif : autonomie et mobilité des données dans les réseaux pair-à-pair.* », Université de Caen Basse-Normandie, 2012.

Par exemple on doit représenter le bon niveau d'incertitude que l'on observe entre un 'V' et un 'X' mal écrits.

Pour obtenir cette représentation doublement optimale, des éléments d'engrammage peuvent disparaître provoquant d'éventuelles recompositions internes, en sous-régions, du partitionnement de la représentation complète.

Nous retrouvions déjà ces principes dans nos travaux, évoqués plus haut, sur les nuées d'oiseaux comme métaphore pour le stockage sécurisé des données dans les réseaux informatiques ouverts:

- migration: les documents (nuées) explorent sans cesse, via leurs fragments (oiseaux), l'espace (réseaux informatiques ouverts) suivant des règles (modélisées par Reynolds et adaptées par nos soins aux réseaux informatiques);
- division cellulaires: Les documents en mouvement (nuées) peuvent se séparer et se reproduire (suivant le principe des codes d'effacement[55]), reconstituant de façon sûre les documents en circulation;
- différenciation: plusieurs documents peuvent partager un même espace (ressources) sans pour autant se dissoudre les uns dans les autres.

Il faut ajouter à cela que des morceaux de documents (oiseaux) peuvent disparaître sans mettre en danger les documents associés; ils peuvent également disparaître pour se redécomposer autrement (évolution des formes).
Ces expériences, qui ont démontré leur viabilité d'approche vis-à-vis des objectifs visés (rendre sûr le stockage de documents dans les réseaux ouverts), sont des tentatives encourageantes vers la production d'une complexité artéfactuelle, dans la mesure où elles sont bio-inspirées.

Conclusion et références

II-E

II-E1) Conclusion

La singularité est co-construite par ses éléments constitutifs (processus autoréférent permanent); elle intègre même des formes à venir, en particulier grâce à la forme canonique associée et à la distance acceptable d'une nouvelle occurrence par rapport à cette forme canonique. Autrement dit toute nouvelle forme à venir qui s'inscrirait dans une «distance» acceptable par rapport à une forme canonique particulière existante, non seulement relèverait de cette forme, mais contribuerait à l'évolution de la dite forme canonique.
La représentation canonique de la lettre 'A' de l'alphabet latin, comme toutes les autres lettres d'ailleurs, englobe non seulement toutes les occurrences de 'A' réalisées, mais aussi toutes celles à venir.

Un ensemble d'instanciations définit une singularité, qui elle-même, dans un processus d'émergence compositionnelle, peut servir à produire d'autres instanciations, lesquelles renforcent/affinent cette singularité. L'environnement qui engendre une ou plusieurs singularités intègre par construction les règles qui définissent/font émerger ces singularités. Il produit implicitement ces règles. Les définir explicitement revient à modéliser partiellement l'environnement. Nous ne sommes plus dans l'environnement, mais «à côté»!
Pour revenir au début de cet article nous dirions que modéliser c'est faire «un pas de côté», nécessaire, mais incomplet.

Faire de l'acquisition de connaissances par des règles combine observation/compréhension et réalisation/ émergence; cela revient d'une certaine façon à s'écarter de l'objet étudié et à ne plus être en capacité de produire implicitement ou explicitement ces règles. C'est ce que disait Brooks à propos de l'usage des symboles dans les programmes d'IA.

Une langue existe, fonctionne, évolue, en un mot «vit» indépendamment de la compréhension que l'on peut en avoir. La compréhension peut être tellement complexe qu'elle est loin de faire l'unanimité chez les experts. C'est aussi vrai pour toutes sortes de connaissances, même scientifiques.

Sur l'exemple de la représentation/reconnaissance des vingt-six lettres de l'alphabet, nous avons tenté de montrer que, suivant le point de vue adopté, on peut décrire un nombre plus ou moins grand (voire infini) de formes qui se rapportent à chaque forme canonique.
Cette modélisation peut devenir fastidieuse. Le challenge consiste à modéliser des environnements, tels que nous les avons esquissés, c.-à-d. capables de produire implicitement ces règles en se basant sur les quatre grands principes de la complexité énoncés plus haut.

La singularité naturelle, comme nous l'avons présentée un peu plus haut, permet des émergences compositionnelles dans les niveaux supérieurs de fonctionnement (super tsunami, fédérations d'états,…) et de représentation (morphème,…).

56 L'informatique opérationnelle est entièrement basée sur ce concept, appelé localement « interface ».

57 Dans « *Parler, c'est tricoter* », aux éditions de l'aube, 2013.

58 Michel Serres parle autrement de la singularité du langage, dans son livre « *L'Incandescent* », p.192-193, en évoquant la notion de « compacité des langues » : « *…Au moment de la catastrophe neuve où s'inventa le sens, ce que nous gagnâmes en précision, nous le perdîmes en convenance. Voilà pourquoi la philosophie montaigne s'élève, unique, parmi des archipels d'évocations récitées, elles-mêmes émergées d'un océan d'halètements. Nous le savons tous : un texte qui seulement énonce, dont la sonorité ni le rythme ne descendent point à cette profondeur, vers ces millions d'années, vers ces neurones aviaires en nous, sonne bavard et nul, ennui et assomme. Écrire ou dire n'ont lieu qu'à saisir d'un coup, à force d'écoute, toute la couche de langue dont l'épaisseur se mesure à partir du sens rare déposé, en haut, sur la chair acoustique, voyelles, rythmes, nombres et mouvements, jusqu'à la base basse où cette clameur touche à la musique souche d'où bifurque en branches l'ensemble des langues… Accéder, sous la signification récente et le chant subtil qui la sculpte, à ce magma compact et archaïque, magnifie toutes les singularités, du français la francité, du portugais la lusitanité, des langues romanes la somme latine, mais débouche sur la bouche d'ombre d'où sort l'ur-musik. De toute œuvre grande émane, comme de cette poche d'ombre, une sorte de mélancolie sonore, une brise calme au ras d'un lac de larmes, l'immense espace du sens. Qui fait ouïr sa rumeur non seulement écrit ou dit, mais commence à parler, comme un ange, la compacité des langues…* »

L'empilement de ces niveaux produit de nouvelles choses et organise celles qui existent.

D'une certaine façon il s'agit de s'abstraire des contingences d'un niveau donné pour déboucher sur d'autres horizons[56].

Cette construction par empilements et compositions successifs de singularités se retrouve partout où le regard et la pensée se portent. Comme le soulignait le peintre Shitao dans son concept aussi bien plastique et sensible que philosophique d'«Unique Trait de Pinceau» (cf. Note 20), ou à sa façon le philosophe Kant dans son concept de jugement réfléchissant, une fois atteint ce niveau singulier de la maîtrise des choses, nul n'est besoin alors de réfléchir, de rationaliser, pour faire un trait unique et multiple à la fois (porteur de tous les sens, un méta-trait intégrant, potentiellement tous les autres traits possibles), ou pour apprécier sans critère de jugement la beauté d'un paysage que nous percevons. Les règles ont été intégrées, encodées… elles sont opérantes.

La singularité est une possibilité parmi bien d'autres, actualisée par des évènements. Les lois ne pouvant qu'exprimer des possibilités et non des certitudes… (I. Prigogine).

Nous conclurons sur ces quelques mots de Claude Hagège: «…Ceci n'est pas un cours, mais permettez-moi de vous rappeler une petite distinction. Distinction que je fais en début d'année aux étudiants qui ont eu la naïveté, la bonté ou l'ingénuité de prendre le cours de linguistique… je précise que le langage est une aptitude définitoire de l'espèce, tandis que les langues en sont les manifestations historiquement et socialement situées dans tel ou tel pays, portées sur leurs fonts baptismaux – et souvent comme des étendards d'affirmation – par telle ou telle nation…»[57]
N'est-ce pas une vision hautement singulière du langage, forme canonique issue d'un processus d'émergence constitutive, en perpétuelle évolution grâce à ses différentes instanciations (les langues) dans des sous-environnement dédiés!…[58]

II-E2) Références

Anthony Gliner, « *La querelle de la camaraderie littéraire. Les romantiques face à leurs contemporains* », éditions Droz, dans la collection « *Histoire des idées et critique littéraire* », 2008.

Benoît Romito, « *Stockage décentralisé adaptatif : autonomie et mobilité des données dans les réseaux pair-à-pair.* », thèse de Doctorat de l'Université de Caen Basse-Normandie en Informatique, 2012.

Catach, « *L'orthographe française* », Paris, éditions Nathan, 1980.

Catherine Fromilhague et Anne Sancier-Chateau, « *Analyses stylistiques. Formes et genres* », aux éditions Nathan Université, 2000.

Charles Pépin, « *Quand la Beauté nous sauve. Comment un paysage ou une œuvre d'art peuvent changer notre vie* », aux éditions Poche Marabout, 2014.

Christian Touratier, « *Les lettres du latin : description sémiologique, fonctionnelle et graphématique* », Université de Provence. http://www.paris-sorbonne.fr/IMG/pdf/Semiologie_fonctionnement__graphematique-2.pdf.

Claude Hagège, « *Parler, c'est tricoter* », aux éditions de l'aube, 2013.

CORBA, « *Common Object Request Broker Architecture* », standard de l'OMG (« *Object Management Group* ») http://www.omg.org/spec/, 1991.

C. Reynolds, « *Flocks, herds and schools : A distributed behavioral model* », in Proceedings of the 14th annual conference on Computer graphics and interactive techniques, SIGGRAPH'87, 1987.

Daniel Calin, « *Quelques repères en psychologie de l'enfant et du préadolescent* », http://dcalin.fr/cerpe/cerpe09.html.

D. Richard, D. Orsal, « *Neurophysiologie, organisation et fonctionnement du système nerveux* », aux éditions Dunod.

François Bourdon, « *Un modèle de dérive des connaissances. Application en bureautique* », thèse de Doctorat de l'Université du Maine, spécialité Intelligence Artificielle, 1992.

François Bourdon, « *Systèmes d'Information Ouverts : Sémantique Interactionnelle des Connaissances et Systèmes Multi-Agents* », rapport d'HDR de l'Université de Caen, spécialité Informatique, 1998.

Gao Xingjian, « *De la création* », aux éditions du Seuil, 2013.

Gavin Pretor-Pinney, « *Le guide du chasseur de nuages* », aux éditions du Points Jean-Claude Lattès, collection Sciences, 2006.

Georges Mounin, « *L'écriture script. Introduction à la sémiologie* », aux éditions de Minuit, 1970.

Havelock, Eric A., « *Aux origines de la civilisation écrite en Occident* », traduit de l'anglais par E. Escobar Moreno, Paris, éditions Maspero, 1981.

« *Henri Matisse, Traits essentiels. Gravures et monotypes 1906-1952* », aux éditions Cabinet des estampes, Genève, 2006.

Hugo Pommier, « *Placement et stockage de l'information bio-inspiré : une approche orientée agent mobile* », thèse de doctorat de l'Université de Caen Basse-Normandie, en Informatique, 2010.

Humberto R. Maturana, Francisco J. Varela, « *L'Arbre de la connaissance. Racines biologiques de la compréhension humaine* », aux éditions Addison-Wesley, 1992.

Ilya Prigogine, « *La fin des certitudes* », aux éditions Odile Jacob, collection Sciences, 1996.

« *Intelligence naturelle et intelligence artificielle* » aux éditions PUF, collection « *psychologie d'aujourd'hui* », 1993.

Jacques Dubois, « *L'institution de la littérature. Introduction à une sociologie (1978)* », Bruxelles, aux éditions Labor-Nathan dans la collection « *Dossiers Média* », 1986.

Jean-Claude Ameisen, « *La sculpture du vivant. Le suicide cellulaire ou la mort créatrice* », aux éditions du Seuil dans la collection Points/Sciences, 2003.

Louis Aragon, « *Aragon, Henri Matisse, roman* », aux éditions Gallimard, dans la collection Quarto, 1998.

Michel Mourlet, « *Le Pic de la Mirandole du XIX^e siècle* », dans « *Littré au XXI^e siècle. Le colloque du bicentenaire* », aux éditions France Univers, 2003.

Michel Serres, « *L'Incandescent* », éditions le Pommier, 2003.

Montaigne, Essais, Livre II, chapitre XVIII, « *Du démentir* ».

Nathalie Heinich, « *L'élite artiste. Excellence et singularité en régime démocratique* », Paris, éditions Gallimard dans la collection « *Bibliothèque des sciences humaines* », 2005.

Pierre Encrevé, « *Soulages, les peintures 1946 - 2006* », aux éditions du Seuil.

Shitao, « *Shitao. Les propos sur la peinture du moine Citrouille-amère* », aux éditions Hermann, 1996.

Simon Singh, « *Le dernier théorème de Fermat* », aux éditions Pluriel, 2010.

« *Stefan Zweig* », Montaigne, PUF, 1935.

Théodore Schwenk, « *Das sensible Chaos/Le Chaos sensible* », deuxième édition française, aux éditions du Centre Triades, 1982.

ÉTUDE THÉORIQUE DU PARTITIONNEMENT DANS LE CADRE DU PARTAGE DE RESSOURCES

III

Le problème étudié ici est celui du placement optimum de ressources en vue du traitement réparti et décentralisé, d'informations dans un environnement ouvert, incertain et à grande échelle, tel que l'apparaît aujourd'hui Internet[1].
L'approche consiste à proposer des algorithmes basés sur un contrôle entièrement décentralisé, dont le principe est l'auto-organisation.
Avant d'aborder les solutions proposées au chapitre IV, nous allons présenter le problème qui servira de fil conducteur au travail de représentation de connaissances, présenté par la suite. Nous mettons l'accent sur la complexité algorithmique de ce problème, dont sa résolution décentralisée ne sera pas uniquement présentée pour mieux coller à la réalité des systèmes visés. Elle apparaîtra comme l'unique façon possible d'aborder ces questions, en particulier à cause de la composante dynamique très forte dans de tels systèmes.
Nous sommes confrontés à une forme de circularité entre la nature des systèmes étudiés, celle des solutions proposées et la réalité des deux. De tels systèmes n'existent que parce qu'ils sont décentralisés…, leurs solutions aussi.
Nous ne perdront pas de vue que ces systèmes fonctionnent à très grande échelle, c.-à-d. qu'ils mettent en jeu simultanément un très grand nombre d'acteurs.

1 On trouve dans la littérature de nombreux travaux qui s'apparentent de près ou de loin à cette problématique. À commencer par le point de Fermat, le problème du cercle minimum, ou encore le problème de Fermat-Torricelli-Steiner dans lequel on cheche un «centre» qui minimise la somme des distances aux points a_i. L'algorithme de Weiszfeld (1937) offre un moyen simple (basé sur une technique de "hill climbing"), à convergence rapide, pour trouver, dans l'espace euclidien, la médiane géométrique d'un ensemble de points. Plus généralement le problème identifié ESFL recherche, pour une configuration topologique donnée, la médiane géométrique pondérée pour des espaces euclidiens de dimension 'd'. Nous verrons plus loin comment notre approche est plus générale et comment elle propose une résolution différente induite par les contraintes auxquelles nous devons faire face dans des situations réelles («naturelles»).

Formalisation du problème à partir de $C_N^K S$

III-A

$SYS_{n,k}$, C_nS_k

Nous appelons un système $SYS_{n,k}$, de type C_nS_k, tout système ouvert, dynamique et décentralisé dans lequel 'n' clients nécessitent simultanément un accès à des ressources au travers d'au plus 'k' serveurs.
Cela pourrait être un prestataire (par exemple de traduction automatique de textes) qui met à disposition de ses clients un ensemble de serveurs sur Internet pour répondre le mieux possible aux besoins changeant des clients, et ce à moindre coût (en optimisant globalement l'usage des ressources en jeu).
Nous nous intéressons donc à des solutions auto-organisées se basant sur les variations des demandes, le nombre des serveurs nécessaires, leur position sur les différents nœuds d'Internet, ainsi que les quantités de ressources dont ces serveurs ont besoin, pour garantir un «bon» fonctionnement global du dispositif.
Dans notre approche, le nombre 'n' de clients fait partie de la définition de notre système $SYS_{n,k}$. En revanche le nombre de serveurs 'k' pourra varier en fonction des solutions apportées à la résolution de $SYS_{n,k}$. Cette étude théorique va nous montrer l'importance de 'k' dans la résolution de $SYS_{n,k}$.
Dans cette première partie nous exposons le problème en nous focalisant sur sa très grande complexité algorithmique induite par la recherche de solutions optimales exactes. Cela nous montre qu'une approche répartie et sans contrôle centralisé semble être la seule approche raisonnable pour aborder ce type de problème. Cela introduit les hypothèses mises en avant dans les parties suivantes et qui sont à la base des heuristiques résolutives proposées.

III-A1) Positionnement du problème $SYS_{N,K}$ et définitions

Un système $SYS_{n,k}$, de type C_nS_k, est défini par l'ensemble CL composé de 'n' clients et l'ensemble SE composé de 'k' serveurs :

CL $CL=\{C_1, C_2, \ldots, C_n\}$, avec $n>0$,
SE $SE=\{S_1, S_2, \ldots, S_k\}$, avec $k>0$.

Les éléments de CL et de SE sont positionnés dans le système $SYS_{n,k}$ via une topologie dans laquelle pourra être mesurée une distance entre eux.

$VOYANT(S_j)$ $\forall S_j \in SE$, $VOYANT(S_j)$ est l'ensemble des clients $C_i \in CL$ qui interagissent avec S_j.
$Card(VOYANT(S_j))$ est le nombre de ces clients.

$VU(C_i)$ $\forall C_i \in CL$, $VU(C_i)=S_j <=> C_i \in VOYANT(S_j)$.

$dem(C_i)$ *Définition* : $dem(C_i)$ est la «demande»[1] (besoin en ressources, mesuré par une valeur entière) de C_i envers le système $SYS_{n,k}$ (via les serveurs); de même $prod(S_j)$ est la quantité (mesurée par une valeur entière) de ressources de $SYS_{n,k}$ nécessaires à S_j pour répondre aux sollicitations des clients dans $SYS_{n,k}$.

$prod(S_j)$

1 Cette demande est une moyenne sur une période suffisamment longue pour être considérée comme constante. Nous aborderons la question de la variabilité de cette valeur, dans les heuristiques proposées dans la suite de ce travail.

Notre approche cherche à trouver des solutions acceptables en terme de production/consommation de ressources, c.-à-d. suivant les trois points de vue suivants:

- individuel: chaque client;
- le prestataire de service: celui qui a la charge des serveurs;
- global ou collectif: celui du partage des ressources dans le système $SYS_{n,k}$.

Cela nécessite le recours à la notion de fonction d'utilité qui se mesure par une ou plusieurs grandeurs quantifiables. Le problème posé ici revient à placer un ou plusieurs serveurs (le nombre fait partie de la solution) sur la grille de telle façon que la somme des productions (18 pour S_2 et 6 pour S_1 sur la figure *Fig-III-A1-1*) soit égale à la somme des demandes (6+6+4+2+2+4 sur la figure *Fig-III-A1-1*) des clients ((*0*)), tout en minimisant une fonction d'utilité qui s'appuiera sur un coût global correspondant à cette solution.

$$\sum_{j=1}^{k} prod(S_j) = \sum_{i=1}^{n} dem(C_i) \qquad (0)$$

Chaque système $SYS_{n,k}$ disposant d'une topologie propre, nous pourrons baser ces fonctions d'utilité sur des métriques associées. Ici nous allons travailler avec des métriques simples qui pourront être remplacées ultérieurement, par des métriques plus adaptées à la nature des problèmes visés. On utilise la distance euclidienne entre les éléments du système. Cette métrique nous permet de travailler aussi bien de façon discrète (positionnement des serveurs exclusivement sur une grille) que continue (positionnement des serveurs sur le plan). La première approche (discrète) permet d'aller plus vite vers la solution optimale, sans nécessairement l'atteindre, la seconde (continue) permet justement, à partir des premiers résultats, d'atteindre la solution optimale. Nous pourrions aussi travailler sur des sphères ou des graphes.

Une autre métrique consiste à travailler à partir du treillis de points, matérialisé par une grille; ceci est plus conforme à un réseau de type Internet.

Définition: $d(C_i,S_j)$ est la distance sur la grille (treillis de points) entre le client C_i et le serveur S_j dans le système $SYS_{n,k}$. — $d(C_i,S_j)$

Cette distance nous permet d'associer une valeur d'utilité globale, appelée CG, à une configuration de clients-serveurs donnée (appelée solution sol_p au problème posé par $SYS_{n,k}$).

Définition: On appelle $sol_{p(n,k)}$, que l'on notera en abrégé par sol_p, une solution au problème posé par $SYS_{n,k}$, qui est constituée de 'n' clients C_i disposés sur la grille et ayant des besoins ($dem(C_i)$), et 'k' serveurs, disposés également sur la grille et mobilisant des ressources $prod(S_j)$ dans $SYS_{n,k}$ en vérifiant l'équation d'équilibre *(0)*. — sol_p

Définition: La fonction d'utilité $CG(sol_p)$, pour une solution sol_p donnée, est égale (*(1)* et *(2)*) à la somme des distances entre chaque client C_i et son serveur S_j «associé» (le plus proche sur la grille) pondérée par le volume du besoin ($dem(C_i)$). — $CG(sol_p)$

CG peut être calculée comme la somme des valeurs locales (cg) propre à chaque serveur S_j :

cg(sol$_p$, S$_j$)

$$cg(sol_p, S_j) = \sum_{i=1}^{Card(VOYANT(S_j))} d(C_i, S_j) \times dem(C_i), \text{ avec } C_i \in VOYANT(S_j) \qquad (1)$$

$$CG(sol_p) = \sum_{l=1}^{k} cg(sol_p, S_l) \qquad (2)$$

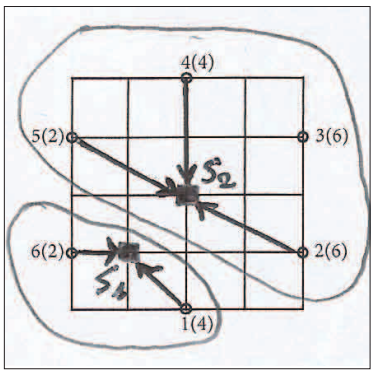


Fig-III-A1-1

Nous allons illustrer ceci sur l'exemple C_6S_2 suivant (cf. la figure *Fig-III-A1-1*) :
CL={C_1,C_2,C_3,C_4,C_5,C_6}, n=6, SE={S_1,S_2}, k=2,
dem(C_2)=dem(C_3)=6, dem(C_1)=dem(C_4)=4, dem(C_5)=dem(C_6)=2,
prod(S_1)=4+2=6, prod(S_2)=6+6+4+2=18, VOYANT(S_1)={C_1,C_6},
VOYANT(S_2)={C_2,C_3,C_4,C_5}, VU(C_2)=VU(C_3)=VU(C_4)=VU(C_5)=S_2,
VU(C_1)=VU(C_6)=S_1, d(C_1,S_1)=2, d(C_6,S_1)=1,
d(C_2,S_2)=d(C_3,S_2)=d(C_5,S_2)=3, d(C_4,S_2)=2,
cg(sol$_p$,S_1)=(4x2)+(2x1)=10,
cg(sol$_p$,S_2)=(6x3)+(6x3)+4x2)+(2x3)=50,
CG(sol$_p$)=10+50=60.

SOL$_{n,k}$

Définition: Nous nommerons SOL$_{n,k}$ l'ensemble des solutions sol$_p$ pour SYS$_{n,k}$.

Nous allons étudier le nombre des solutions possibles sol$_p$ (Card(SOL$_{n,k}$)), afin de mesurer la complexité en temps pour trouver la (ou les) solution(s) optimale(s) (sol$_p^*$), c.-à-d. celle(s) qui minimise(nt) la fonction d'utilité CG.

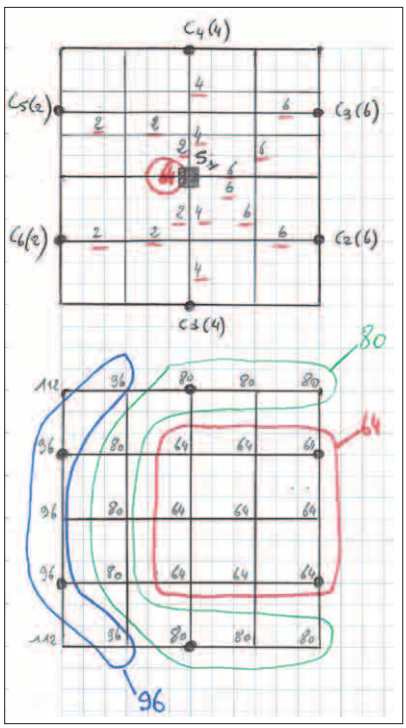


Fig-III-A1-2

CAS PARTICULIER AVEC UN SEUL SERVEUR (K=1)

Dans la mesure où la grille (carrée de côté 'D') possède D^2 positions possibles pour le serveur S_j, en première approximation, il y aura autant de solutions sol$_p$ à SYS$_{n,k}$ que de positions sur la grille. C'est le cas de la figure *Fig-III-A1-2* avec un seul serveur S_j (k=1), où nous obtenons : Card(SOL$_{n,k}$)=D^2. *(2b)*

Cette formule représente un cas particulier (k=1) d'une formule plus générale que nous verrons par la suite.
Toujours sur cet exemple C_6S_1 (cf. la figure *Fig-III-A1-2*), l'aspect symétrique de la solution du problème (positionnement sur la grille des clients et choix des valeurs demandées par les clients) simplifie sa compréhension, mais n'est absolument pas une condition pour sa résolution.
Sur chaque point de la grille nous avons calculé *(2)* le coût global CG(sol$_p$) de la solution sol$_p$ avec un seul serveur et six clients. Chaque valeur sur la grille correspond donc au coût obtenu pour la solution (satisfaire tous les clients) si le serveur se trouve sur le point en question.

CG(sol$_p$)=cg(sol$_p$,S_1)=64 si S_1 est sur l'un des neuf points de la grille entourés de rouge (cf. la figure *Fig-III-A1-2*).
On voit très nettement se dégager des zones géographiques allant de la moins coûteuse (64) pour la zone cerclée de rouge, à la plus coûteuse (112) pour les points extrêmes nord-ouest et sud-ouest de la grille.

CAS GÉNÉRAL AVEC K SERVEURS (K>1)

Prenons C_6S_2 (k=2) et l'une des solutions sol_p (cf. la figure *Fig-III-A1-1*) de $SYS_{6,2}$, avec deux serveurs $S_1(6)$ et $S_2(18)$.
Nous avons :

$CG(sol_1)=cg(sol_1,S_1)+cg(sol_1,S_2)$.

Or il existe de nombreuses façons de positionner S_1 et S_2 sur la grille, et, de façon concomitante, de répartir les clients sur chacun des serveurs. Prenons simplement les deux solutions suivantes sol_1 (cf. la figure *Fig-III-A1-2b-1)* et sol_2 (cf. la figure *Fig-III-A1-2b-2*).

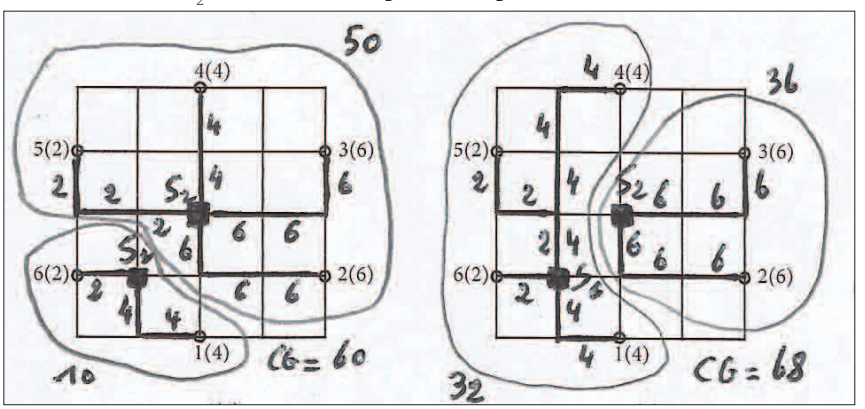


Fig-III-A1-2b-1 *Fig-III-A1-2b-2*

Dans les deux cas les serveurs S_1 et S_2 occupent les mêmes positions sur la grille, bien qu'ils n'obtiennent pas les mêmes valeurs pour le coût global CG des solutions sol_1 et sol_2 correspondantes. Pour sol_1 (cf. la figure *Fig-III-A1-2b-1*), nous avons :

$cg(sol_1,S_1)=(2x4)+(1x2)=10$,
$cg(sol_1,S_2)=(3x6)+(3x6)+(3x2)+(2x4)=50$, soit
$CG(sol_1)=10+50=60$.

Pour sol_2 (cf. la figure *Fig-III-A1-2b-2*), nous obtenons :

$cg(sol_2,S_1)=32$, $cg(sol_2,S_2)=36$, soit $CG(sol_2)=32+36=68$.

Propriété: La répartition des clients sur les serveurs conduit à des solutions sol_p différentes.

L'exemple suivant sol_3 (cf. la figure *Fig-III-A1-2c*) montre clairement le rôle de la position des serveurs sur la grille dans la recherche du coût optimal. Dans cet exemple nous avons choisi une répartition des clients qui est particulièrement avantageuse, comme en témoigne la valeur de $CG(sol_3)=20+20=40$, si on la compare aux deux premières solutions évoquées.

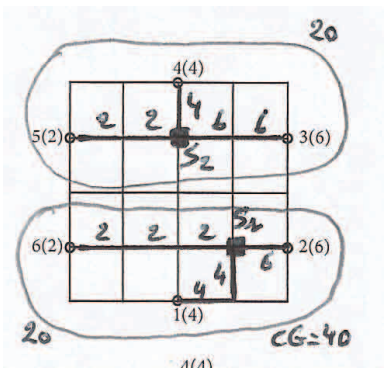


Fig-III-A1-2c

Il nous faut donc revoir la formule du calcul du nombre de solutions sol_p en y intégrant la possibilité d'avoir plusieurs serveurs (k>1).
En première approximation, c.-à-d. en gardant les répétitions (deux serveurs peuvent occuper la même position sur la grille) et les doublons, et avec ordre entre les serveurs, nous obtenons la nouvelle formule suivante :

$$Card(SOL_{n,k}) = \underbrace{D^2 D^2 ... D^2}_{\text{k fois}} = D^{2\times k} \qquad (3)$$

Comme nous venons de le voir au travers de quelques exemples, les solutions sol_p de $SYS_{n,k}$ dépendent aussi de la répartition des clients vis-à-vis des serveurs. Pour affiner *(3)* nous allons introduire la notion de configuration ($conf_1$) dans l'espace des solutions défini par $SOL_{n,k}$.
Reprenons C_6S_k dans lequel nous avons six clients et 'k' serveurs. Nous allons dénombrer les différentes répartitions des clients. Fixons k=1; dans ce cas la répartition des clients est unique. Ils interagissent tous avec l'unique serveur. Nous revenons à *(3)*. Si k=2, il nous faut dénombrer le nombre de répartitions des six clients sur deux serveurs.

$conf_1$ *Définition*: On appelle configuration $conf_1$ la l^e répartition des 'n' clients sur les 'k' serveurs dans $SYS_{n,k}$.

$Conf_{q(n,k)}$ *Définition*: On nomme $Conf_{q(n,k)}$ l'ensemble des répartitions possibles $Conf_1$ des 'n' clients sur les 'k' serveurs dans $SYS_{n,k}$.

$Conf_{i1,i2,\ldots,ij}$ *Définition:* On nomme $Conf_{i1,i2,\ldots,ij}$ l'ensemble des configurations $conf_1 \in Conf_{q(n,k)}$ tel que i_1 clients sont répartis sur le serveur S_1, i_2 clients sur S_2,…, i_j clients sur S_j avec $i_1+i_2+\ldots+i_j=n$.

Dans le cas de C_6S_2, où l'on cherche à répartir six clients sur deux serveurs, nous avons:
$Conf_{q(6,2)}=\{Conf_{1,5}\}\cup\{Conf_{2,4}\}\cup\{Conf_{3,3}\}$.
Et dans le cas de C_6S_3 (avec trois serveurs) nous avons:
$Conf_{q(6,3)}=\{Conf_{1,1,4}\}\cup\{Conf_{1,2,3}\}\cup\{Conf_{2,2,2}\}$.

Il est à noter que dans le cas particulier où k=2, le nombre d'ensembles $Conf_{i,j}$ est de n/2 si 'n' est pair et (n-1)/2 si 'n' est impair. Pour n=6 nous avons bien trois sous-ensembles $Conf_{1,5}$, $Conf_{2,4}$ et $Conf_{3,3}$.

Cette notion de configuration ($conf_1$) introduit la notion fondamentale de partitionnement des clients vis-à-vis des serveurs.
Nous pouvons maintenant énoncer une nouvelle formule *(3b)* pour le nombre de solutions $SOL_{n,k}$ au problème posé par $SYS_{n,k}$, toujours en gardant les répétitions:

$$Card(SOL_{n,k}) = D^{2k} \times Card(Conf_{q(n,k)}) \qquad (3b)$$

Cette formule intègre pour chaque position des serveurs le nombre d'arrangements des clients sur ces serveurs.
Sur des exemples simples nous énumèrerons toutes les solutions de toutes les configurations possibles et mettrons en évidence la solution optimale pour chaque configuration de $Conf_{q(n,k)}$.

$VAL_{n,k}$ *Définition*: Soit $VAL_{n,k}$ l'ensemble des valeurs $CG(sol_p)$ des solutions $sol_p \in SOL_{n,k}$, nous avons $Card(VAL_{n,k}) \leq Card(SOL_{n,k})$.

$CG^*_{n,k}$ *Définition*: Nous appelons $CG^*(Conf_{q(n,k)})$, que l'on notera aussi $CG^*_{n,k}$, la valeur CG de la solution optimale (sol_p^*) du système $SYS_{n,k}$, c.-à-d. la borne inférieure de toutes les valeurs des différentes solutions sol_p de $SOL_{n,k}$, dans toutes les configurations $Conf_{i1,i2,\ldots,ij}$.

$$CG^*(Conf_{q(n,k)}) = CG^*_{n,k} = \min(VAL_{n,k}) \qquad (4)$$

Nous verrons plusieurs solutions pour définir $Conf_{q(n,k)}$ dont la formule la plus rigoureuse, mais pas la plus intéressante sur le plan calculatoire (complexité temporelle hors de portée) sera celle du nombre de Stirling de 2^{e} espèce.

Nous pouvons obtenir la solution optimale (de valeur $CG^{*}_{n,k}$) pour $SYS_{n,k}$ soit en cherchant la valeur minimale CG parmi toutes les solutions sol_{p} (dans $VAL_{n,k}$), c'est la formule (4), soit en cherchant l'optimale par répartition $CG^{*}(Conf_{i1,i2,\ldots,ij})$ et en gardant la meilleure de toutes.

Nous allons reprendre l'exemple fil rouge C_6S_j.

La grille a un côté de valeur 5 (D=5), le nombre de clients est 6 (n=6), le nombre de serveurs 'k' est variable (k=1→solution (a), k=2→solution (b), k=3→solution (c)).

Si l'on se base sur *(3b)* pour le calcul du nombre des solutions $Card(SOL_{n,k})$, nous obtenons :

(a) $SE=\{S_1\}$, $k=1 \Rightarrow Card(SOL_{6,1})=5^2=25$,
puisque $Card(Conf_{q(6,1)})=1$,
$SOL_{6,1}=\{sol_1, sol_2, \ldots, sol_{25}\}$,
$SOL_{6,1}=\{(pos_1,(conf_1)),(pos_2,(conf_1)), \ldots,(pos_{25},(conf_1))\}$;

(b) $SE=\{S_1,S_2\}$, $k=2 \Rightarrow$
$Card(SOL_{n,k})=5^{2\times2}\times Card(Conf_{q(6,2)})=625\times31=19375$,
puisque $Card(Conf_{q(6,2)})=Card(Conf_{1,5})+Card(Conf_{2,4})+Card(Conf_{3,3})=6+15+10=31$,
$SOL_{6,2}=\{sol_1, sol_2, \ldots, sol_{19375}\}$,
$SOL_{6,2}=\{(pos_1,pos_2,(conf_1)),\ldots,(pos_1,pos_2,(conf_{31})),$
$(pos_1,pos_3,(conf_1)),\ldots,(pos_1,pos_3,(conf_{31})),$
$\ldots,$
$(pos_{24},pos_{25},(conf_1)),\ldots,(pos_{24},pos_{25},(conf_{31}))\}$;

(c) $SE=\{S_1,S_2,S_3\}$, $k=3 \Rightarrow$
$Card(SOL_{n,k})=5^{2\times3}\times Card(Conf_{q(6,3)})=15625\times90=1406250$,
car $Card(Conf_{q(6,3)})=Card(Conf_{1,1,4})+Card(Conf_{1,2,3})+Card(Conf_{2,2,2})=15+60+15=90$,
$SOL_{6,3}=\{sol_1, sol_2, \ldots, sol_{1406250}\}$.

Définition: pos_i correspond au point P_i de coordonnées (x_i/y_i) sur la grille du système $SYS_{n,k}$.

pos_i

Nous allons préciser la valeur du nombre de configurations possibles $Conf_{q(n,k)}$ pour tout 'n' entier strictement positif et tout 'k' entier strictement positif.

Sur la figure *Fig-III-A1-2*, avec un seul serveur (cas (a)) nous avons $VAL_{n,k}=\{64,80,96,112\}$ pour vingt-cinq solutions/positions possibles pour $SYS_{n,k}$. Dans ce cas $CG^{*}_{n,k}=64$.

Pour des questions de symétrie, mais pas uniquement, la cardinalité de $VAL_{n,k}$ est inférieure à celle de $SOL_{n,k}$, le nombre de solutions de $SYS_{n,k}$.

On le voit bien sur la figure *Fig-III-A1-2* avec les régions qui regroupent des points de la grille qui possèdent la même valeur $CG(sol_p)$ pour des solutions sol_p pourtant différentes les unes des autres. Elle peut, dans certains cas, lui être très inférieure ($Card(VAL_{n,k}) << Card(SOL_{n,k})$). Autrement dit beaucoup de solutions peuvent être redondantes dans la mesure où elles conduisent au même résultat (CG). Avant d'explorer ce problème dans toutes ses dimensions, nous allons revenir sur le coût du calcul de la solution optimale ($CG^*_{n,k}$).

III-A2) Coût du calcul de la solution optimale ($CG^*_{n,k}$)

Revenons sur le calcul du nombre de solutions dans $SYS_{n,k}$. Nous sommes toujours avec une grille de côté 'D', qui possède D^2 positions possibles accessibles à chaque serveur. La formule *(3)* nous donne le nombre de positions avec ordre et avec répétitions.

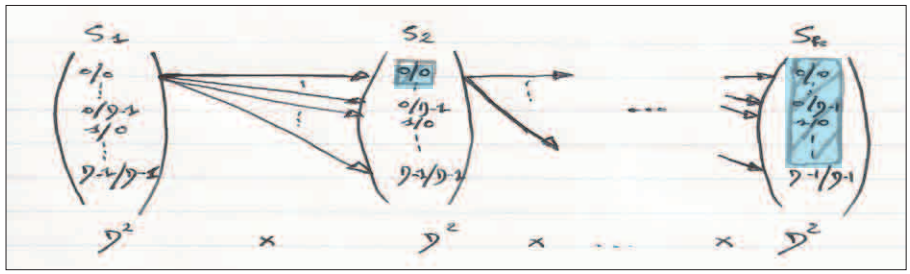


Fig-III-A2-3

MM_{k-p} *k-position*

Définition: Nous appelons multi-points MM_{k-p} de $SYS_{n,k}$, le vecteur constitué de la *k-position* $(pos_{i1}, pos_{i2}, \ldots, pos_{ik})$, représentant les différentes positions (pos_j) des 'k' serveurs sur la grille ($j \in [1,k]$).

On pourra noter $MM_{k-p}=\{P_1, P_2 \ldots, P_k\}$, avec P_j de position pos_j, soit (x_j, y_j).

POS_k

Définition: Nous posons POS_k, l'ensemble des multi-points MM_{k-p} de $SYS_{n,k}$ ($p \in [1, Card(POS_k)]$).

Nous pouvons réécrire *(3b)*:

$$Card(SOL_{n,k}) = Card(POS_k) \times Card(Conf_{q(n,k)}), \quad (5)$$

avec, d'après (3), $Card(POS_k) = D^{2k}$.

En réalité nous ne souhaitons pas garder les répétitions (deux points identiques ne peuvent appartenir à un multi-points MM_{k-p}) et nous ne souhaitons pas considérer l'ordre des serveurs ($\{P_i, P_j\} \Leftrightarrow \{P_j, P_i\}$) dans un multi-points MM_{k-p}.

Petit rappel sur la notion d'arrangement :

Définition : On appelle arrangement de 'p' objets toutes suites ordonnées de 'p' objets pris parmi les 'n' objets. Le nombre d'arrangements de 'p' objets pris parmi 'n' est noté A_n^p.

- dans le cas d'arrangements ordonnés (12≠21) avec répétition (11, 22,… comptent), on a :

$$A_n^p = n^p. \qquad (6)$$

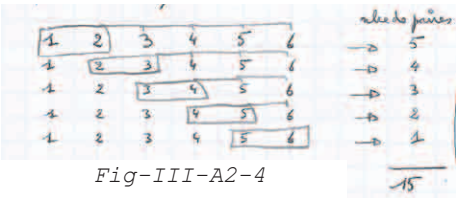


Fig-III-A2-4

En effet, si n=6 et p=2 on a $A_6^2=6^2=36=15+15+6$;
dû à l'ordre inverse
dû aux répétitions (11,22,33,44,55,66)

- si l'on retire les répétitions, on obtient :
  $A_n^p=n!/(n-p)!$, avec $0<p\leq n$. *(6a)*
  Ici $A_6^2=6!/4!=30$;

- enfin si l'on ne garde que les paires sans ordre (23⇔32,…) la formule devient :
  $A_n^p=n!/p!(n-p)!$, avec $1<p\leq n$, *(6b)*
  ici $A_6^2=6!/2!4!=15$.

Dans notre cas il s'agit de trouver les arrangements A_n^p en utilisant la formule *(6b)*, avec $n=D^2$ et $p=k$. Nous obtenons donc la formule *(7)* pour le nombre de multi-points ($Card(POS_k)$) de $SYS_{n,k}$:

$$Card(POS_k) = \frac{D^2!}{k!(D^2-k)!} \qquad (7)$$

La formule *(5)* se réécrit de la façon suivante *(7b)* :

$$Card(SOL_{n,k}) = \frac{D^2!}{k!(D^2-k)!} \times Card(Conf_{q(n,k)}) \qquad (7b)$$

Si $k=D^2$, ce qui revient à positionner un serveur sur chaque point de la grille, alors nous obtenons pour $Card(POS_k)$ une seule solution (car 0!=1), la plus coûteuse en nombre de serveurs et la plus efficace en terme d'échange d'informations sur les réseaux. À l'opposé, si k=1 nous retrouvons D^2 solutions pour $Card(POS_k)$, une sur chaque point de la grille. Là encore cette situation n'est pas réaliste car centralisée.
Entre ces deux extrêmes, la formule *(7)* nous montre le rôle très important que va jouer le nombre de serveurs 'k' dans la complexité de la recherche de la solution optimale. Nous étudierons cela plus en détails ultérieurement, sur des situations concrètes.
La complexité de *(7)* dépend de 'k'.

$Card(POS_1)_6$ =36
$Card(POS_2)_6$ =630
$Card(POS_3)_6$ =7140
$Card(POS_4)_6$ =58905
$Card(POS_5)_6$ =376992
$Card(POS_6)_6$ =1947792
$Card(POS_7)_6$ =8347680
$Card(POS_8)_6$ =30260340
$Card(POS_9)_6$ =94143280
$Card(POS_{10})_6$ =254186856
$Card(POS_{11})_6$ =600805296
$Card(POS_{12})_6$ =1251677700
$Card(POS_{13})_6$ =2310789600
$Card(POS_{14})_6$ =3796297200
$Card(POS_{15})_6$ =5567902560
$Card(POS_{16})_6$ =7307872110
$Card(POS_{17})_6$ =8597496600
$Card(POS_{18})_6$ =9075135300
$Card(POS_{19})_6$ =8597496600
$Card(POS_{20})_6$ =7307872110
$Card(POS_{21})_6$ =5567902560
$Card(POS_{22})_6$ =3796297200
$Card(POS_{23})_6$ =2310789600
$Card(POS_{24})_6$ =1251677700
$Card(POS_{25})_6$ =600805296
$Card(POS_{26})_6$ =254186856
$Card(POS_{27})_6$ =94143280
$Card(POS_{28})_6$ =30260340
$Card(POS_{29})_6$ =8347680
$Card(POS_{30})_6$ =1947792
$Card(POS_{31})_6$ =376992
$Card(POS_{32})_6$ =58905
$Card(POS_{33})_6$ =7140
$Card(POS_{34})_6$ =630
$Card(POS_{35})_6$ =36
$Card(POS_{36})_6$ =1

$Card(POS_1)_5$ =25
$Card(POS_2)_5$ =300
$Card(POS_3)_5$ =2300
$Card(POS_4)_5$ =12650
$Card(POS_5)_5$ =53130
$Card(POS_6)_5$ =177100
$Card(POS_7)_5$ =480700
$Card(POS_8)_5$ =1081575
$Card(POS_9)_5$ =2042975
$Card(POS_{10})_5$ =3268760
$Card(POS_{11})_5$ =4457400
$Card(POS_{12})_5$ =5200300
$Card(POS_{13})_5$ =5200300
$Card(POS_{14})_5$ =4457400
$Card(POS_{15})_5$ =3268760
$Card(POS_{16})_5$ =2042975
$Card(POS_{17})_5$ =1081575
$Card(POS_{18})_5$ =480700
$Card(POS_{19})_5$ =177100
$Card(POS_{20})_5$ =53130
$Card(POS_{21})_5$ =12650
$Card(POS_{22})_5$ =2300
$Card(POS_{23})_5$ =300
$Card(POS_{24})_5$ =25
$Card(POS_{25})_5$ =1

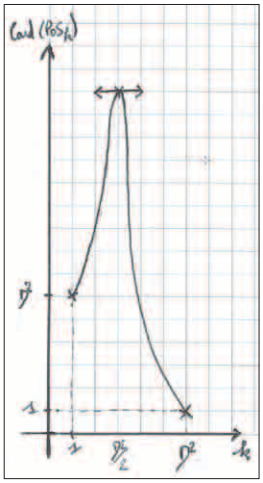


Fig-III-A2-4b

En réalité le nombre de positions ($Card(POS_k)$) n'est pas compris entre 1 et D^2. La courbe symétrique possède un point d'inflexion (extremum horizontal) pour $k=D^2/2$ si D^2 est pair, et deux points d'inflexion de même valeur, respectivement $k_1=(D^2-1)/2$ et $k_2=(D^2+1)/2$, si D^2 est impair, lui faisant atteindre, dans les deux cas, des valeurs extrêmement élevées (cf. la figure *Fig-III-A2-4b*). En effet si D=6, nous aurons l'extremum pour $k=D^2/2=18$, avec $Card(POS_{18})=9\ 075\ 135\ 300$.

Ceci signifie que pour D=6 (grille de trente-six points), nous avons $Card(POSk) \in [1, 9\ 075\ 135\ 300]$, $\forall k \in [1, D^2]$.

Chaque solution sol_p au problème posé par $SYS_{n,k}$, revient à positionner le ou les serveurs sur la grille et à répartir l'ensemble des clients dans un partitionnement $conf_p$ donné. Le coût de chaque solution sol_p nécessite pour chaque client le calcul de sa distance au serveur correspondant, multiplié par sa demande *(1)*.

$\theta_1(SOL_{n,k})$

Définition: On appelle $\theta_1(SOL_{n,k})$ la complexité, en temps, du calcul de l'ensemble des solutions sol_p incluses dans $SOL_{n,k}$, et $\theta_1(CG(sol_p))$ la complexité pour calculer CG (cf. la figure *Fig-III-A2-4c*) dont la formule est donnée par *(2)*.

$$\theta_1(SOL_{n,k}) = Card(SOL_{n,k}) \times \theta_1(CG(sol_p)) \quad (8)$$

Si l'on veut maintenant calculer le coût de l'obtention de la solution optimale $CG^*_{n,k}$, il faut ajouter le coût du tri de toutes les solutions obtenues.

Dans la suite de l'exposé, nous allons montrer qu'aux dimensions réelles de systèmes comme Internet, le calcul centralisé de la solution optimale est tout simplement irréaliste et hors de portée pour longtemps.

Pour cela nous allons préciser exactement ce que vaut $Card(Conf_{q(n,k)})$ dans *(5)* et étudier comment cette grandeur peut être approximée, nous conduisant à des solutions sous-optimales. Tout le travail consistera alors à montrer que ces solutions restent raisonnables/acceptables. Pour cela nous tablerons sur le fait que les solutions proposées s'appuieront sur les contraintes/caractéristiques du système $SYS_{n,k}$ étudié.

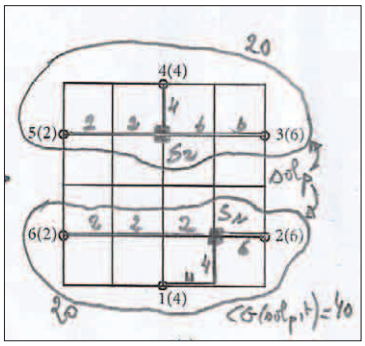


Fig-III-A2-4c

Une approche consiste alors à utiliser les résultats des études théoriques sur les solutions optimales exactes, pour mieux comprendre la nature profonde du problème et en tirer partie dans les méthodes/heuristiques de recherche des solutions pour des dimensions du problème, a priori, inaccessibles.

Au vue des premiers éléments de l'analyse du problème, l'une des hypothèses fortes avancées, pour casser la complexité, mais aussi aborder une résolution décentralisée du problème, est de partitionner l'espace des clients en limitant ainsi le nombre de clients par partition, avec un serveur par partition. On calculera alors l'ensemble des solutions pour un partitionnement donné ($Conf_{q(n,k)}$, avec 'n' clients et 'k' serveurs).

Nous allons commencer cette étude empirique sur l'exemple C_6S_j, pour en extraire des lois plus générales comme celle qu'il existe entre la complexité du calcul des solutions ($\theta_1(SOL_{n,k})$) et la valeur ($CG^*_{n,k}$) du coût de la solution optimale obtenue pour $SYS_{n,k}$ dans toutes les configurations $Conf_{i1,i2,\ldots,ij}$ *(4)*. On verra qu'il n'existe pas de configuration idéale, mais un continuum entre des configurations qui possèdent des qualités et des défauts contradictoires. Aux deux extrêmités on trouvera la situation qui consiste à faire peu de partitions et celle, au contraire, qui en fera beaucoup. La première ne sera pas cher à calculer, pour un résultat médiocre en terme de coût global de fonctionnement, alors que la seconde inversera cette tendance.

Les choses ne sont pas linéaires, car nous verrons qu'au-delà d'un seuil, le partitionnement augmentant, il devient contre-productif. Le cas caricatural consiste à tellement partitionner le système que l'on met un serveur par client. Dans ce cas on peut dire que le coût de calcul de la solution optimale tend vers zéro. On peut même ajouter que la valeur obtenue pour $CG^*_{n,k}$ tend aussi vers zéro, ce qui est l'optimal absolu. Sauf que cette situation, qui reviendrait à superposer les notions de serveur et de client au point de les confondre, est irréaliste, car contraire au principe même de la nature du système $SYS_{n,k}$ (ici le paradigme clients/serveurs). Ces considérations montrent qu'il est important de bien connaître les lois qui peuvent régirent la recherche de solutions, afin de faire reposer les méthodes heuristiques sur une connaissance précise du terrain.

III-A3) Première étude de la loi entre $Card(SOL_{n,k})$ et $CG^*_{n,k}$

Rappelons qu'une configuration donnée $conf_q$, pour résoudre $SYS_{n,k}$, consiste à mobiliser un nombre donné de serveurs qui seront positionnés sur la grille. Le nombre des clients, leur position sur la grille et leur besoin en ressources dans $SYS_{n,k}$, sont fixés au départ. Pour cette étude nous partons de l'hypothèse que les besoins des clients sont constants (discrétisés par unité de temps). Dans notre proposition de résolution du problème C_nS_k, ces besoins pourrons évoluer dans le temps suivant des logiques propres aux clients. Nous pourrons ainsi proposer des heuristiques de résolution, basées sur une capacité d'auto-ajustement à des variations non-déterministes de l'environnement…

Une configuration donnée $Conf_1$ consiste donc à fixer un nombre de partitions avec un serveur par partition. La valeur $CG^*(Conf_{q(n,k)})$ obtenue sera la valeur minimale des valeurs de toutes les solutions contenues dans cette configuration *(4)*; alors que la valeur optimale CG^* du système $SYS_{n,k}$ sera la valeur optimale de toutes les valeurs optimales dans chacune des configurations possibles.
Comme nous l'avons vu dans le calcul du nombre de solutions $Card(SOL_{n,k})$ de $SYS_{n,k}$ dans (*7b*), la répartition des clients dans chacune des partitions ($Conf_1$) de $Conf_{q(n,k)}$ y joue un rôle important.

Les configurations de $SYS_{n,k}$

Nous sommes confrontés au problème, bien connu en analyse combinatoire, du *k-partitionnement*. Le nombre de Stirling de 2e espèce, appelé S(n,k) [Contet,T2, p38] calcule de façon exacte les nombres de *k-partitions* dans un ensemble 'n'; le nombre de Bell, $\varpi(n)=\Sigma_k$, définit le nombre de Stirling sur toutes les configurations possibles ($Conf_{q(n,k)}$).

Nous avons vu plus haut que $conf_1$ correspond à la 1e répartition des 'n' clients sur 'k' serveurs dans $SYS_{n,k}$.

k-partition

Définition: Nous nommons *k-partition* d'un ensemble de 'n' éléments, la configuration $conf_1$ des 'n' éléments dans 'k' sous-ensembles, appelés partitions de cet ensemble et notées $Part_i$ avec $i\in[1,Card(Conf_{q(n,k)})]$.

Propriété: Une *k-partition* forme un partitionnement de l'ensemble des 'n' éléments en 'k' partitions disjointes deux à deux et dont l'union forme l'ensemble des 'n' éléments tout entier.

$conf_1=\{Part_1,Part_2,\ldots,Part_k\}_1$ est une *k-partition* sur $SYS_{n,k}$.

Sur l'exemple C_6S_j, ou plus généralement C_nS_k, avec $0\leq k\leq n$ où 'n' est le nombre de clients et 'k' le nombre de serveurs, nous cherchons à exprimer le nombre total des *k-partitions* dans l'ensemble des 'n' clients. Dans un premier temps fixons l'étude C_6S_k avec k=2. Nous cherchons à calculer le nombre de possibilités de répartir les six clients dans des *2-partitions*.
Nous avons trois possibilités (a,b,c):

a) un client pour la partition $Part_1$ et cinq clients pour $Part_2$, que nous notons $Conf_{1,5}$;
b) deux clients pour $Part_1$ et quatre clients pour $Part_2$, que nous notons $Conf_{2,4}$;
c) trois clients pour $Part_1$ et trois clients pour $Part_2$, que nous notons $Conf_{3,3}$.

Dans notre contexte nous nous intéressons aux arrangements sans ordre ni répétition. En appliquant *(6b)*, nous avons:
$Card(Conf_{1,5})=6!/1!(6-1)!=6$ arrangements, (a)
$Card(Conf_{2,4})=6!/2!(6-2)!=15$ arrangements, (b)
$Card(Conf_{3,3})=6!/3!(6-3)!=20$ arrangements, (c)
avec,
$Card(Conf_{q(6,2)})=Card(Conf_{1,5})+Card(Conf_{2,4})+Card(Conf_{3,3})$ et
$Card(Conf_{q(6,2)})=6+15+20=41$.

Nous allons énumérer ces configurations:

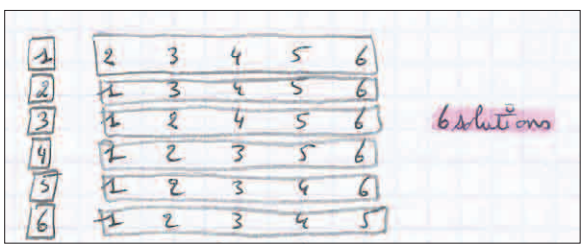


Fig-III-A3-5

a) $Conf_{1,5}$: six *2-partitions*:

$conf_1=\{(1),(2,3,4,5,6)\}=\{Part_1,Part_2\}_1$,
$conf_2=\{(2),(1,3,4,5,6)\}=\{Part_1,Part_2\}_2$,
$conf_3=\{(3),(1,2,4,5,6)\}=\{Part_1,Part_2\}_3$,
$conf_4=\{(4),(1,2,3,5,6)\}=\{Part_1,Part_2\}_4$,
$conf_5=\{(5),(1,2,3,4,6)\}=\{Part_1,Part_2\}_5$,
$conf_6=\{(6),(1,2,3,4,5)\}=\{Part_1,Part_2\}_6$.

b) $Conf_{2,4}$: quinze *2-partitions ;*

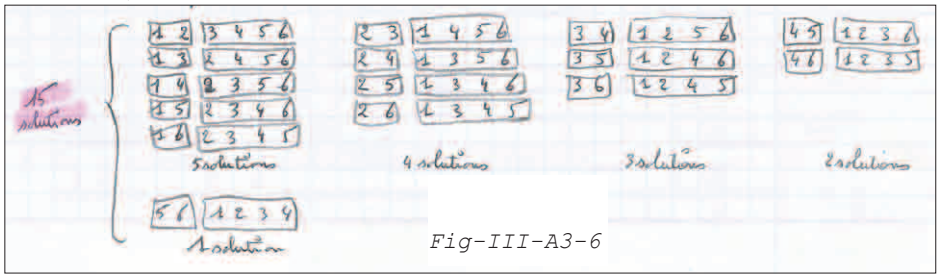


Fig-III-A3-6

c) $Conf_{3,3}$: vingt *2-partitions*:

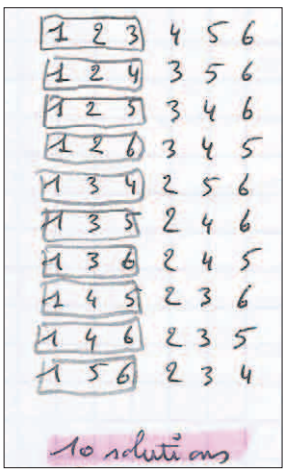


Fig-III-A3-7

$conf_1=\{(1,2,3),(4,5,6)\}=\{Part_1,Part_2\}_1$,
$conf_2=\{(1,2,4),(3,5,6)\}=\{Part_1,Part_2\}_2$,
$conf_3=\{(1,2,3),(3,4,6)\}=\{Part_1,Part_2\}_3$,
$conf_4=\{(1,2,6),(3,4,5)\}=\{Part_1,Part_2\}_4$,
$conf_5=\{(1,3,4),(2,5,6)\}=\{Part_1,Part_2\}_5$,
$conf_6=\{(1,3,5),(2,4,6)\}=\{Part_1,Part_2\}_6$,
$conf_7=\{(1,3,6),(2,4,5)\}=\{Part_1,Part_2\}_7$,
$conf_8=\{(1,4,5),(2,3,6)\}=\{Part_1,Part_2\}_8$,
$conf_9=\{(1,4,6),(2,3,5)\}=\{Part_1,Part_2\}_9$,
$conf_{10}=\{(1,5,6),(2,3,4)\}=\{Part_1,Part_2\}_{10}$.

En utilisant la formule *(6b)*, sans répétition et sans ordre, le nombre d'arrangements de trois parmi six (A_6^3) est de vingt.
Sur les vingt arrangements obtenus seuls dix sont différents (cf. la figure *Fig-III-A3-7*).
Cette formule ((A_n^p) *(6b)*) n'est donc pas bonne pour calculer les *k-partitions* puisqu'elle compte des arrangements en doublons qui peuvent devenir très nombreux.

Le bon calcul, pour notre exemple C_6S_2 de *2-partitions*, est de trente-et-un (6+15+10) arrangements différents et non quarante-et-un. Or il existe une autre formule, celle appelé du nombre de Stirling de 2e espèce (S(n,k)), qui donne les bonnes valeurs (sans répétition, sans ordre et sans doublons).

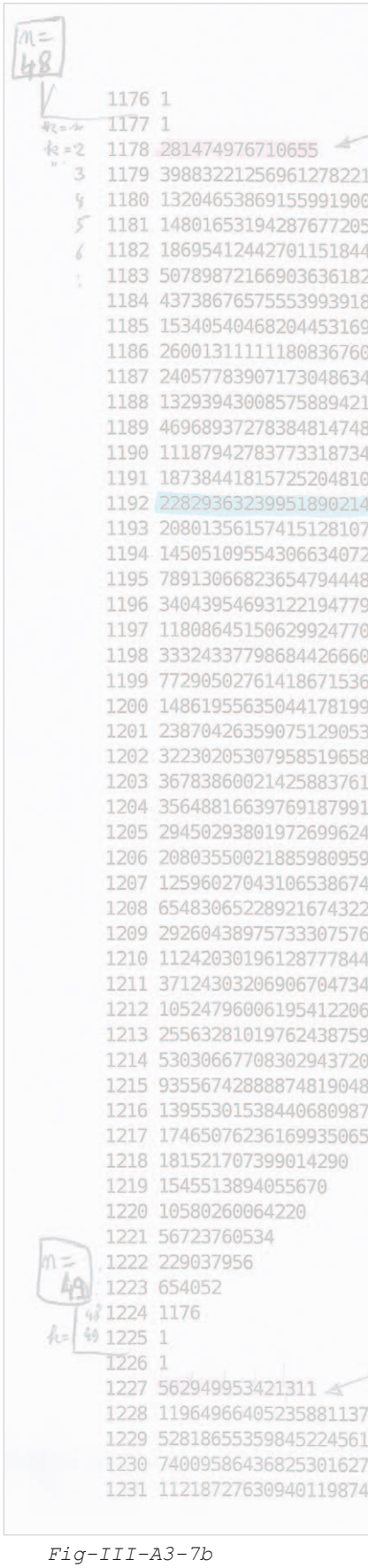


Fig-III-A3-7b

S(n,k)

Petit rappel sur le nombre de Stirling de 2e espèce :

S(n,k) est le nombre de *k-partitions* d'un ensemble N à 'n' éléments.
S(n,k)>0 si 1≤k≤n et S(n,k)=0 si 1≤n<k.
On pose S(0,0)=1 et S(0,k)=0, si k≥1, nous obtenons :

$$S(n,k) = \frac{1}{k!}\sum_{0\le j\le k}(-1)^j\binom{k}{j}(k-j)^n \qquad (9)$$

avec

$$\binom{k}{j} = \frac{k!}{(k-j)!j!}, etj \le k \qquad (10) \quad \text{et } j\le k.$$

C'est le principe fondamental du dénombrement qui postule que si une expérience se compose de deux phases avec 'k' résultats possibles en phase-1 et 'n' en phase-2, pour chacun des résultats de la première phase il y a au total k×n résultats de l'expérience.
Tout sous-ensemble de 'j' objets choisis sans répétition dans un ensemble 'k', noté $\binom{k}{j}$, représente les combinaisons de 'j' objets pris parmi les 'k'.

Les nombres S(n,k) apparaissent comme les fonctions symétriques monomiales de degré (n-k) des 'k' premiers entiers [Contet,T2, p.42]. On peut écrire le nombre de Stirling de 2e espèce, qui est le nombre de partitions d'un ensemble à 'n' éléments en 'p' parties, de la façon suivante :

$$\left\{ \begin{matrix} n \\ p \end{matrix} \right\} = \left\{ \begin{matrix} n-1 \\ p-1 \end{matrix} \right\} + p \times \left\{ \begin{matrix} n-1 \\ p \end{matrix} \right\} \qquad (11)$$

La formule exacte de S(n,k) devient alors :

$$S(n,k) = \sum_{C_1+C_2+\ldots+C_k=n-k} 1^{C_1}2^{C_2}\ldots k^{C_k} \qquad (12)$$

avec la relation de récurrence suivante :

$$S(n+1,k)=k\times S(n,k)+S(n,k-1) \qquad (13)$$

Si l'on prend l'exemple S(5,2) des *2-partitions* avec cinq objets, et celui S(6,2) avec six objets, nous avons n=5, k=2 et n-k=3, respectivement n=6, k=2 et n-k=4.

$$S(5,2) = \sum_{C_1+C_2=3} 1^{C_1}2^{C_2} \qquad (14)$$

$$S(6,2) = \sum_{C_1+C_2=4} 1^{C_1}2^{C_2} \qquad (15)$$

$S(5,2)=1^3x2^0+1^2x2^1+1^1x2^2+1^0x2^3=15$ et
$S(6,2)=1^4x2^0+1^3x2^1+1^2x2^2+1^1x2^3+1^0x2^4=31$.

Ces sommes consistent à additionner toutes les combinaisons possibles de (C_1,C_2) telle que $C_1+C_2=3$ dans le cas S(5,2) et $C_1+C_2=4$ pour S(6,2).

Nous obtenons bien les valeurs obtenues exhaustivement pour Card($Conf_{q(5,2)}$) et Card($Conf_{q(6,2)}$) cette fois sans répétition, sans ordre, ni redondance.
Grâce à *(12)* nous avons les résultats suivants (en nombre de solutions) :

S(6,3)=90, // six clients et trois serveurs
S(15,2)=16383, // quinze clients et deux serveurs
S(15,3)=2 375 101, // quinze clients et trois serveurs
S(15,4)=42 355 950. // quinze clients et quatre serveurs

On peut aussi appliquer la relation de récurrence *(13)* :

S(6,2)=2xS(5,2)+S(5,1)=2x15+1=31.

Trente-et-un étant le nombre de *2-partitions* avec six éléments, c.-à-d. $Conf_{1,5}$, $Conf_{2,4}$ et $Conf_{3,3}$.

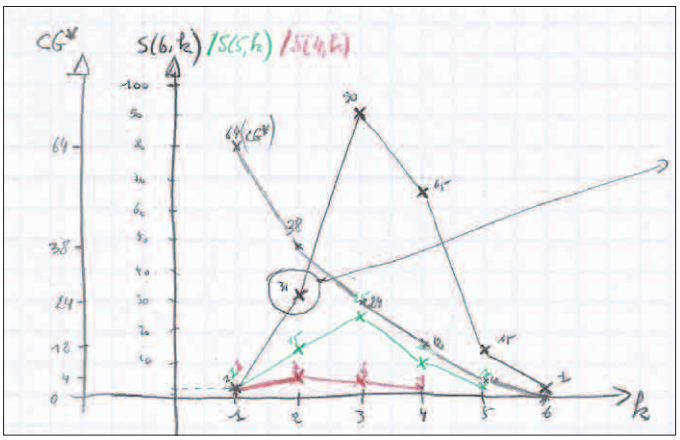


Fig-III-A3-8

Sur la figure *Fig-III-A3-8* nous voyons S(n,k) en faisant varier 'k' (en abscisse) pour trois valeurs 'n' (en ordonnée : n=6 en noir, n=5 en vert et n=4 en rouge). Nous avons superposé $CG^*_{n,k}$ avec S(n,k) en fonction de 'k'. Nous allons affiner l'analyse de cette superposition.
On voit que $CG^*_{n,k}$ décroit lorsque 'k' est croissant. Dans le même temps le nombre de *k-partitions* (S(n,k)) suit une courbe qui possède un point d'inflexion. Il est des valeurs de 'k' pour lesquelles les calculs de S(n,k) seront beaucoup plus nombreux.

Stirling 2e espèce S(n,k), le cas particulier où k=2

Le cas particulier des *2-partitions* (k=2) dans lequel notre problème est décomposé en deux partitions, pourra avoir de l'importance pour les heuristiques à venir, surtout dans le cas d'algorithmes dichotomiques comme celui de la mitose cellulaire, que nous proposerons.

La formule de récurrence *(13)* nous donne :
S(n+1,k)=kxS(n,k)+S(n,k-1), $\forall n>0$,
$\Rightarrow$S(n,2)=2S(n-1,2)+S(n-1,1), or $\forall m\in N^*$ et m>0,
S(m,1)=1 d'où S(n,2)=2xS(n-1,2)+1,
$\Rightarrow$d'après (15) $S(n,2)=2^0+2^1+\ldots+2^{n-2}=S(n-1,2)+2^{n-2}$.
S(n,2) est égal à la somme cumulée des puissances de 2 de 0 à (n-2), soit $2^{n-1}-1$.

$$S(n,2)=2^{n-1}-1 \qquad (16)$$

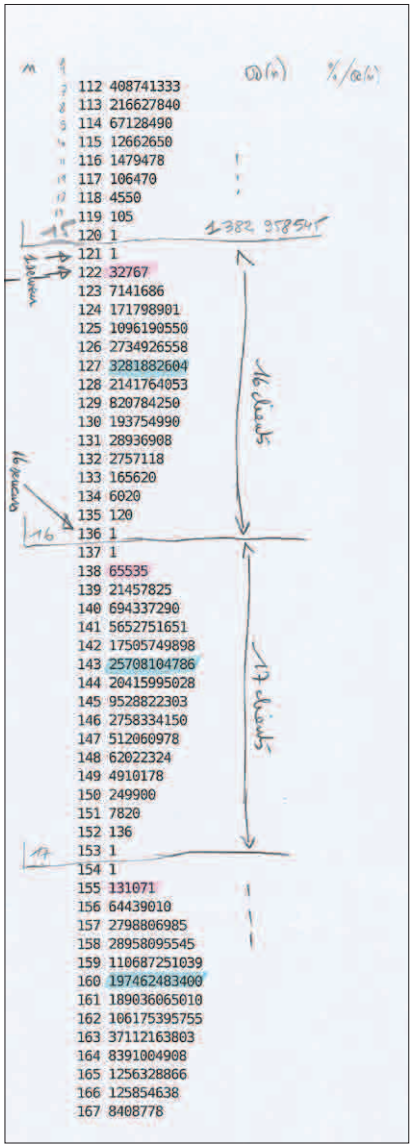


Fig-III-A3-8b

La formule (5) nous indique que le nombre de solutions sol_p de $SOL_{n,k}$ ($Card(SOL_{n,k})$) est égale au produit du nombre de *k-positions (multi-points MM_{k-p})* appelé $Card(POS_k)$ par le nombre de *k-partitions* appelé $Card(Conf_{q(n,k)})$:

$$SOL_{n,k}(Card(SOL_{n,k}))=Card(POS_k)\,x\,Card(Conf_{q(n,k)}).$$

Pour mieux comprendre la complexité en temps du problème C_nS_k il nous faut reprendre ces formules *(5)*, *(7)* et *(7b)* en les appliquant à des situations concrètes.

En partant d'une grille de trente-six points (6x6) (D=6). Nous pouvons mettre au plus un client par point de la grille, donc trente-six clients au plus ($n\in[1,36]$):

- si n=36, le nombre de *2-partitions* est :
$Card(Conf_{q(36,2)})=S(36,2)=34\ 359\ 738\ 367$,
et celui des *2-positions* :
$Card(POS_2)=36!/2!(36-2)!=630$.
On obtient :
$Card(SOL_{36,2})=21\ 646\ 635\ 171\ 210$;
- si n=6 toujours avec une grille de trente-six points (6X6), et toujours loin d'une situation réaliste sur Internet, voici les valeurs obtenues en fonction de 'k' :
$Card(Conf_{q(6,k)})_{k=1\ldots6}=S(6,k)_{k=1\ldots6}=\{1,31,90,65,15,1\}$,
$Card(POS_k)_{k=1\ldots6}=\{36,630,7140,58905,376992,1947792\}$,
$Card(SOL_{6,k})_{k=1\ldots6}=\{36,19530,642600,3828825,5654880,1947792\}$;
- en prenant un exemple un peu plus grand : la grille fait cent points (10x10) (D=10) et nous avons seize clients (n=16)(figure *Fig-III-A3-8b*). Dans ce cas nous avons $Card(POS_k)_{k=1\ldots100}\in[50,126410606327829]$, avec la valeur maximale pour $D^2/2=50$. Nous allons regarder les cinq premières valeurs (k=1…5) du calcul :
$S(16,k)_{k=1\ldots5}=\{1,32767,7141686,171798901,1096190550\}$,
$Card(POS_k)_{k=1\ldots5}=\{100,4950,161700,3921225,75287520\}$,
$Card(SOL_{16,k})_{k=1\ldots5}=\{100,162196650,1154850626200,673662145573725,8.25x10^{16}\}$.

Au-delà du fait que les nombres sont très grands et qu'ils se multiplient ($SOL_{n,k}(Card(SOL_{n,k}))=Card(POS_k)\,x\,Card(Conf_{q(n,k)})$), il sera intéressant de comparer les grandeurs entre elles. C'est ce que nous ferons plus loin. Néanmoins il nous faut contextualiser ces chiffres. En effet le nombre de *k-partitions* dépend de 'n' et de 'k', mais ne dépend pas de la grille (D^2), alors que le nombre de *k-positions* dépend de 'k' et de la grille (D^2), mais pas de 'n'.

Nous avons vu au départ que 'n', le nombre de clients, est fixé dans $SYS_{n,k}$, alors que 'k' fait partie de la résolution du problème et peut être variable. On peut dire également que la topologie de la grille et ses dimensions sont fixées dans $SYS_{n,k}$ et resteront constantes pendant sa résolution.

Nous verrons plus loin, dans notre tentative de représentation des connaissances avec un système C_nS_k, que le choix de 'n' et de la grille (topologie et dimension) pourront prendre en compte la nature des connaissances à représenter.

Si nous reprenons nos trois configurations $Conf_{1,5}$, $Conf_{2,4}$ et $Conf_{3,3}$ (a, b et c), regardons leur contextualisation (cas discret) :

a) $Conf_{1,5}$, avec six *2-partitions* ;

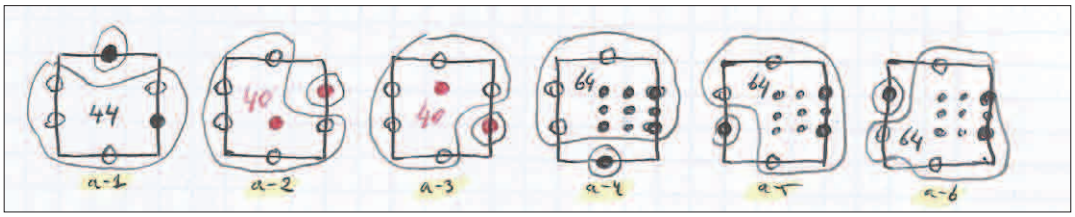


Fig-III-A3-9

b) $Conf_{2,4}$, avec quinze *2-partitions* ;

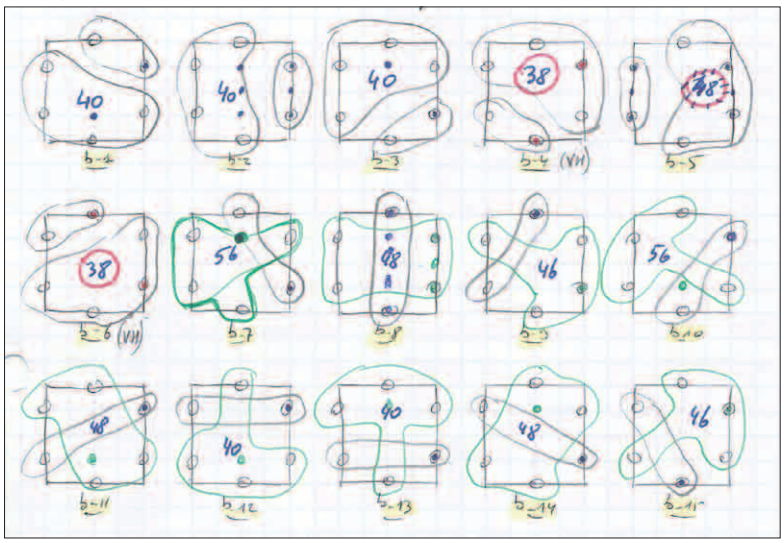


Fig-III-A3-10

c) $Conf_{3,3}$, avec vingt *2-partitions*.

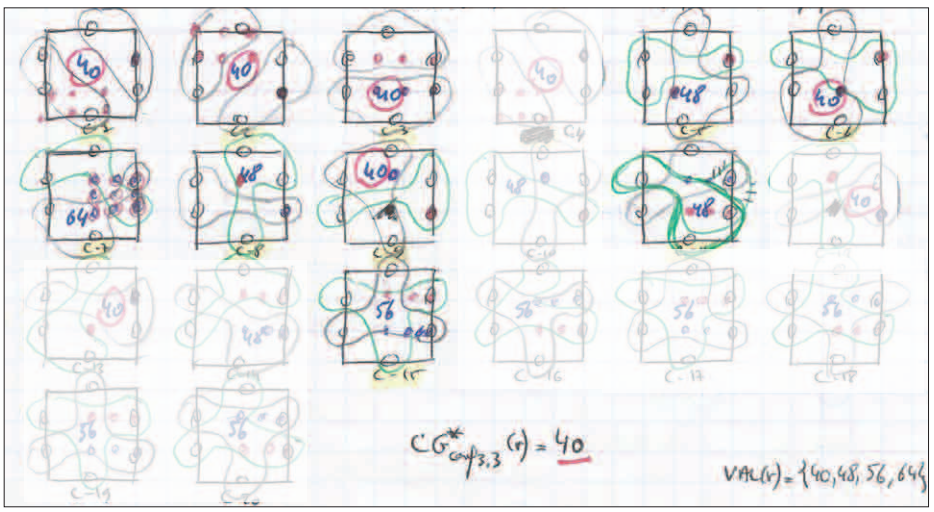


Fig-III-A3-11

Nous avons matérialisé sur ces schémas les valeurs $CG(sol_p)$ *(2)* pour chaque *2-partition*, nous permettant de voir la valeur optimale $CG^*(Conf_{q(n,k)})$ *(4)* pour la configuration $Conf_{q(6,2)}$. L'idée étant de regarder le rôle de la topologie d'une *2-partition* dans les résultats obtenus.

Par exemple nous observons que pour $Conf_{1,5}$ (cf. la figure *Fig-III-A3-9*) toutes les solutions (partitionnement en *2-partitions* ($Part_1$,$Part_2$)) sont topologiquement disjointes (les enveloppes convexes des partitions $Part_1$ et $Part_2$ sont disjointes entre elles).

En regardant $Conf_{2,4}$ (cf. la figure *Fig-III-A3-10*) nous remarquons que de b-1 à b-6 les partitions $Part_1$ et $Part_2$ sont également topologiquement disjointes, ce qui n'est pas le cas pour les autres *2-partitions* (de b-7 à b-15). Sur cet exemple les *2-partitions* optimales sont topologiquement disjointes.
Enfin pour $Conf_{3,3}$ (cf. la figure *Fig-III-A3-11*) nous avons représenté les vingt *2-partitions* possibles, dont dix redondantes (grisées). Dans ce cas il faut être plus nuancé. Parmi les solutions optimales (CG=40) la plupart sont topologiquement disjointes, mais les autres ne le sont pas.

Pour le même exemple si l'on prend trois partitions ($Part_1$, $Part_2$ et $Part_3$) au lieu de deux, nous obtenons les résultats suivants :

$CG^*(Conf_{1,1,4})=28$, avec trente solutions, dont quinze différentes,
$CG^*(Conf_{1,2,3})=26$, avec soixante solutions,
$CG^*(Conf_{2,2,2})=24$, avec quatre-vingt-dix solutions, dont quinze différentes.

Avec trois partitions $CG^*(Conf_{q(6,3)})=24$, alors que la valeur était de 38 pour deux partitions et 64 pour une seule (cf. la figure *Fig-III-A1-2)*.

Nous voyons se dessiner l'amorce d'une loi entre $CG^*_{n,k}$ et le nombre de *k-partitions*, et par conséquence S(n,k).
Dans la configuration $Conf_{1,1,4}$ avec trois partitions, nous voyons (cf. la figure *Fig-III-A3-12*) qu'il suffit d'annuler le coût de la paire de la configuration $Conf_{2,4}$ à deux partitions pour obtenir ce résultat. Nous verrons effectivement que plus le nombre de partitions augmente et plus $CG^*_{n,k}$ diminue. C'est la façon dont la diminution opère qui va nous intéresser.
On observe également sur d'autres exemples que plus les partitions sont homogènes et plus la *k-partition* optimale est bonne.

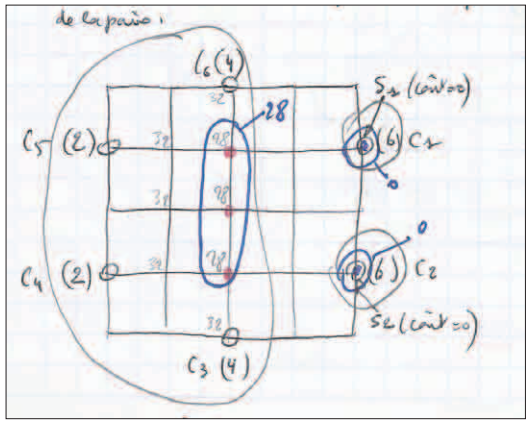


Fig-III-A3-12

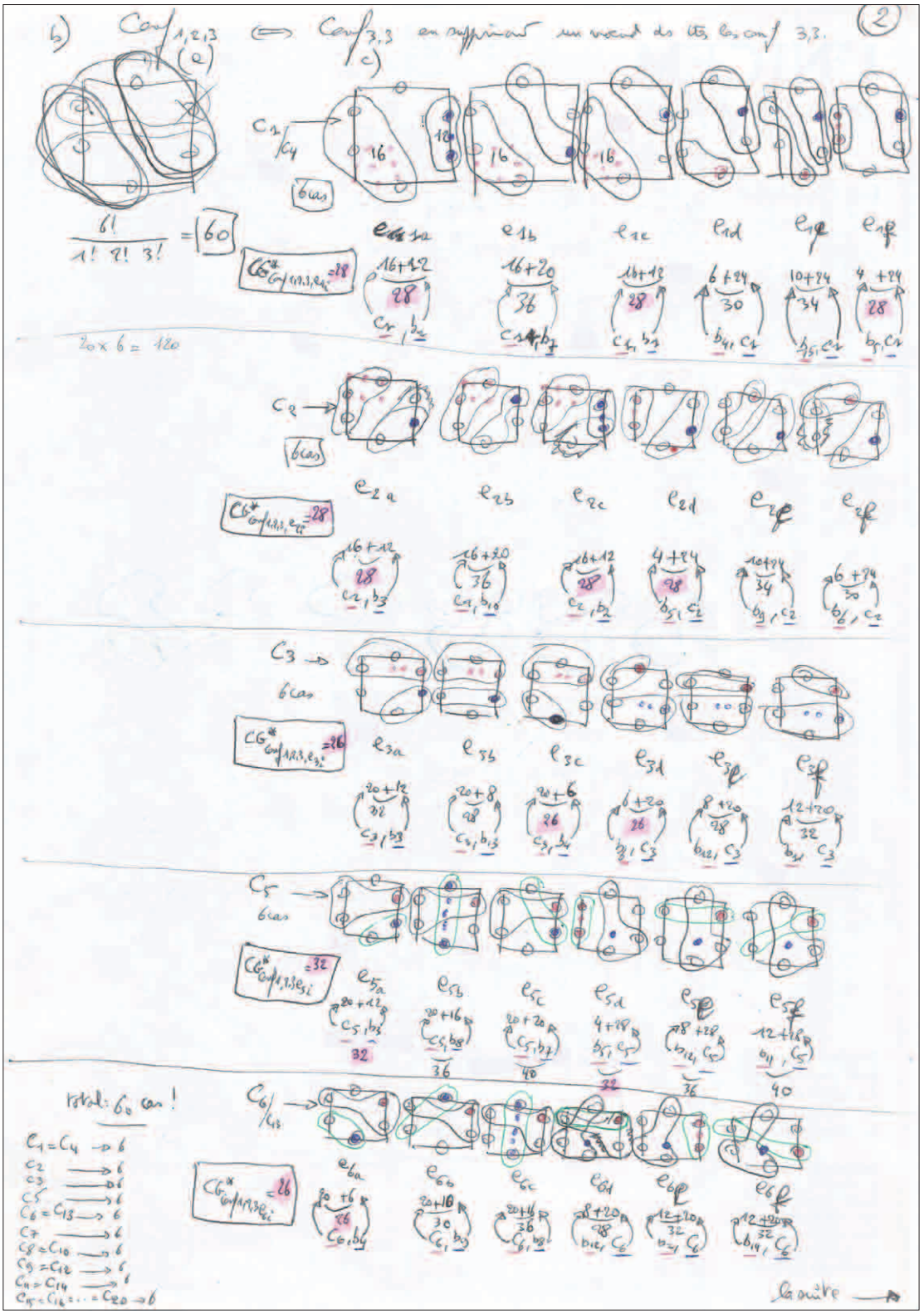


Fig-III-A3-12b

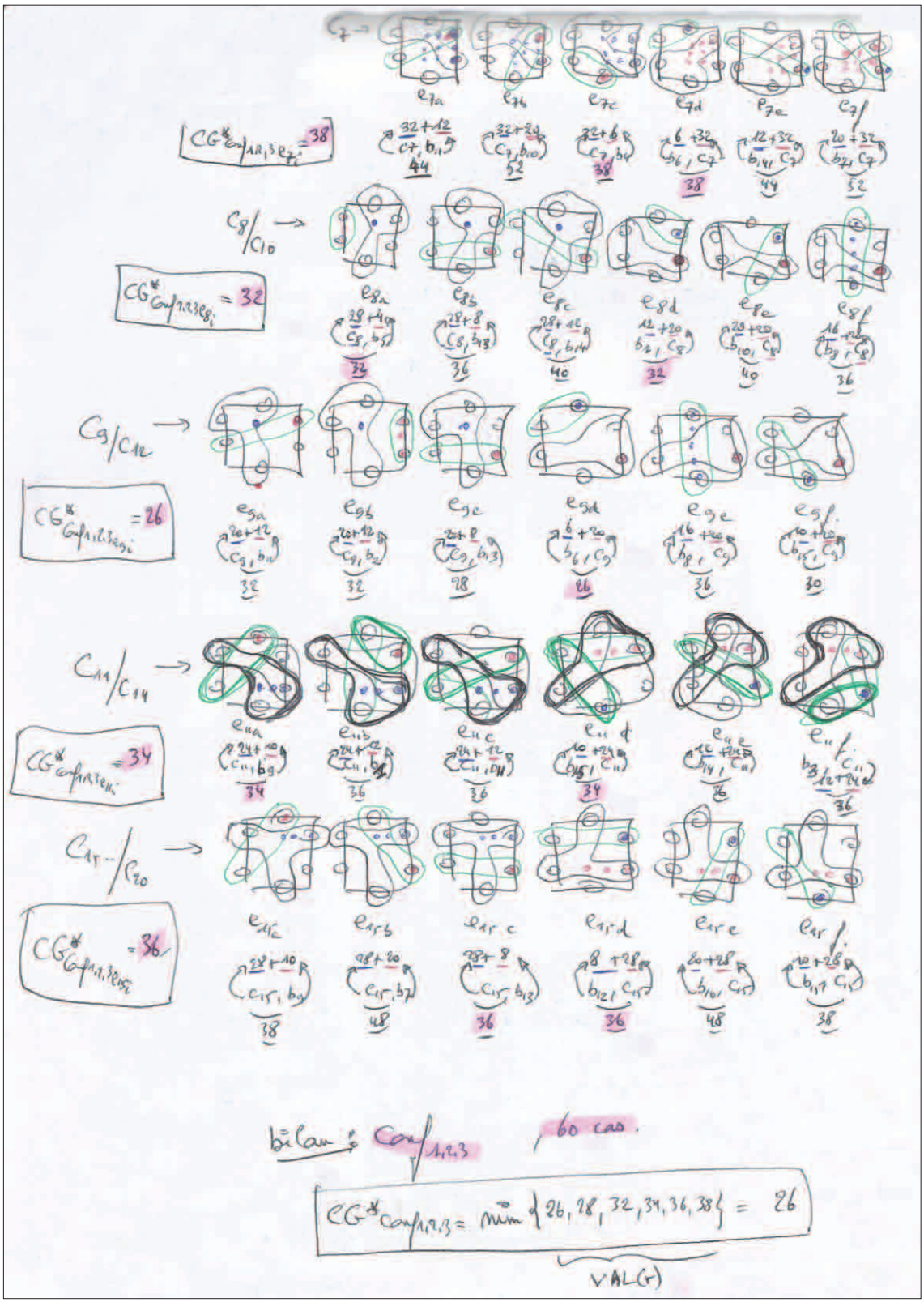


Fig-III-A3-12c

Fig-III-A3-12d

Nombre de Bell $\varpi(n)$

On ne peut pas parler du nombre de Stirling de 2e espèce sans parler du nombre de Bell[1] $\varpi(n)$, dans lequel il s'inscrit. D'autant que dans notre approche heuristique de la résolution du problème $SYS_{n,k}$ nous regarderons les solutions $SOL_{n,k}$ en faisant varier 'k'. En effet nous proposeront des algorithmes auto-adaptatifs avec lesquels les serveurs pourront être créés dynamiquement ('k' variable), jouant ainsi un rôle décisif dans l'auto-organisation du système de résolution du problème $SYS_{n,k}$.

$$\varpi(n) = \sum_{k=1}^{n} S(n,k) \qquad (17)$$

$$\varpi(6) = \sum_{k=1}^{6} S(6,k) \qquad (18)$$

Ce nombre correspond au nombre de partitions possibles d'un ensemble de 'n' éléments en sommant les *1-partitions*, les *2-partitions*,… les *k-partitions*.

Par exemple

$\varpi(6)=S(6,1)+S(6,2)+S(6,3)+S(6,4)+S(6,5)+S(6,6)$,

ce qui donne :

$\varpi(6)=1+31+90+65+15+1=203$.

Puisque le nombre de Bell s'appuie sur S(n,k), il comprend toutes les configurations possibles sans répétition, ni ordre, ni redondance. Il est intéressant de noter sur cet exemple que pour l'intervalle ((n/2)±1) on couvre plus de 91% des solutions de $\varpi(n)$.

Pour conclure cette première approche de la loi entre $Card(SOL_{n,k})$ et $CG^{*}_{n,k}$, nous avons vu, sur la figure *Fig-III-A3-8,* la situation respectivement pour six, cinq et quatre partitions (k=6,5,4). On voit très nettement se dessiner une loi qui sera confortée avec les calculs exacts et exhaustifs. Plus on partitionne ($k \rightarrow n$) et plus le coût optimal induit ($CG^{*}_{n,k}$) diminue ($CG^{*}_{n,k}=0$ si k=n). Ceci n'est pas étonnant car plus on met de serveurs et moins les clients en seront éloignés, minimisant ainsi les coûts liés à l'accès aux ressources de $SYS_{n,k}$. Cette approche n'est pas réaliste car coûteuse en infrastructure réseau, peu évolutive et sans mutualisation possible.

En revanche, ce qui est plus étonnant, c'est que le nombre de configurations pour calculer les différentes valeurs de $CG^{*}_{n,k}$ en fonction de 'k' suivent la courbe du nombre de Stirling de 2e espèce (par le nombre de *k-positions*) avec un pic médian en terme de coût de calculs à effectuer. Mettre un seul serveur (ou peu) sera facile à calculer mais pas plus réaliste (fragilité au fonctionnement, aux attaques…).
Cela conduira à rechercher un partitionnement homogène, efficace, adaptatif, avec des serveurs équilibrés en charge, donc interchangeables et bien répartis.

1 Si 'X' est une variable aléatoire suivant une distribution de Poisson avec une moyenne 'λ', alors son n^e moment est $E(X^n)=\Sigma_{k=1}{}^{n}S(n,k)\lambda^k$. En particulier le n^e moment d'une distribution de Poisson de moyenne 1 est précisément le nombre de partitions d'un ensemble de taille 'n', qui est le nombre de Bell (formule de Dobinski).

Le nombre de Stirling de 2e espèce S(n,k) nous permet de calculer toutes les configurations des *k-partitions* parmi un nombre de 'n' objets (ici des clients) sans redondance. Les valeurs obtenues sont très vite inutilisables, d'autant que la complexité de recherche des solutions optimales pour $SYS_{n,k}$ dépend de la dimension de la grille (positions possibles des serveurs) et du nombre 'k' de serveurs (partitions). Nous allons regarder s'il est possible d'encadrer S(n,k), sans dénaturer notre problème. Pour cela nous allons revenir rapidement sur la borne supérieure de S(n,k) que nous appellerons $\theta_{MAX}(n,k)$, avant de chercher une borne inférieure qui nous intéresse particulièrement et que nous nommerons $\psi(n,k)$.

III-A4) L'APPROXIMATION $\theta_{MAX}(N,K)$

Pour calculer le nombre de partitions de 'n' objets en 'k' groupes disjoints de taille respectivement $n_1,n_2,\ldots,n_k$, avec $n_1,n_2,\ldots,n_k$ des entiers non négatifs et $n=n_1+n_2+\ldots+n_k$, nous obtenons la formule *(19)* suivante:

$$\theta_{MAX}(n,k)_{n_1,n_2,\ldots,n_k} = \binom{n}{n_1,n_2,\ldots,n_k} = \frac{n!}{n_1!n_2!\ldots n_k!} \qquad (19)$$

$\theta_{MAX}(n,k)_{n1,n2,\ldots,nk}$ concerne un seul ensemble de configuration $Ci(n_1,n_2,\ldots,n_k)$ telle que $n=n_{i1}+n_{i2}+\ldots+n_{ik}$, or il existe plusieurs configurations qui vérifient cette propriété.

Pour $\theta_{MAX}(5,3)$ on a les deux ensembles de configurations C1(1,1,3) et C2(1,2,2), respectivement un objet, un objet et trois objets, et un objet, deux objets et deux objets.
Soit $conf_q(5,3)$ l'ensemble des configurations pour n=5 et k=3, alors on a:
$conf_q(5,3)=\{C1\}\cup\{C2\}$,
$\theta_{MAX}(5,3)=\theta_{MAX}(5,3)_{C1}+\theta_{MAX}(5,3)_{C2}$
$=(5!/1!1!3!)+(5!/1!2!2!)=20+30$,
$\theta_{MAX}(5,3)=50$.

Pour $\theta_{MAX}(5,3)_{C1}$ on observe dix configurations redondantes (en jaune sur la figure *Fig-III-A4-13*), auxquelles il faut ajouter quinze pour $\theta_{MAX}(5,3)_{C2}$. Cela fait en tout vingt-cinq configurations redondantes sur les cinquante obtenues en tout.
On retrouve bien les vingt-cinq cas sans redondance liés à la formule de Stirling (S(5,3)).
À cause des redondances nous avons:
$\forall n>2,\ \theta_{MAX}(n,3)>S(n,3)$.

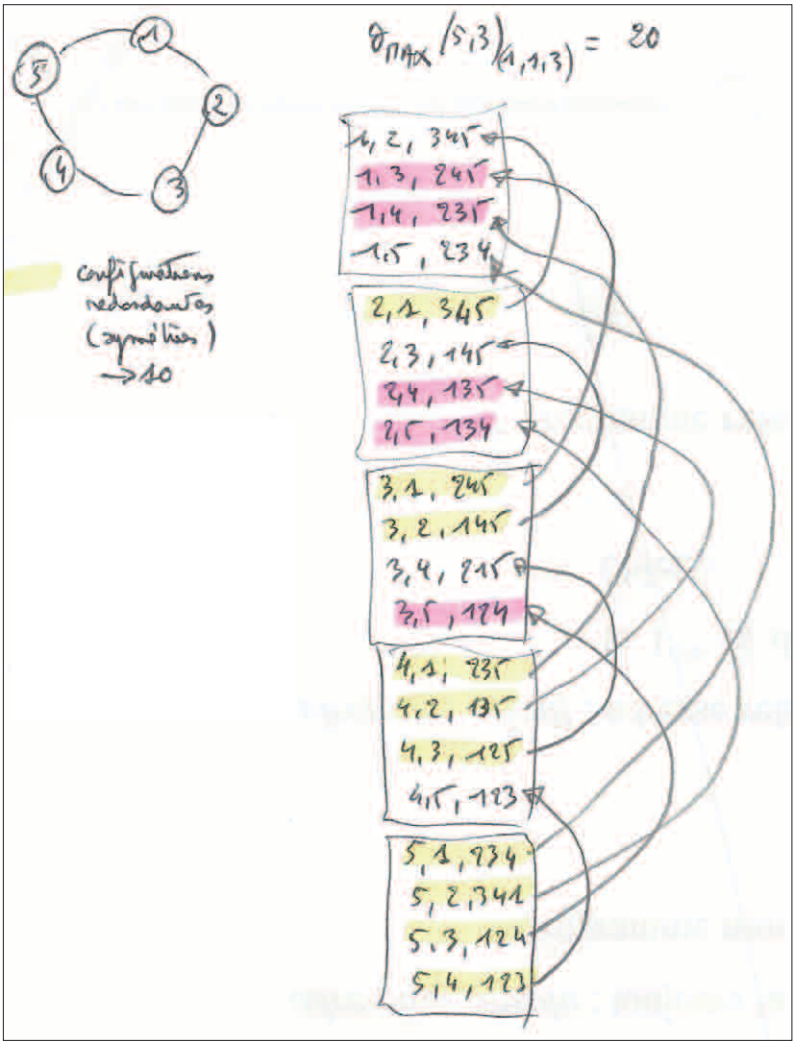


Fig-III-A4-13

Dans le cas particulier $\theta_{MAX}(n,2)$, avec deux serveurs (k=2), on obtient la propriété suivante qui dépend de la parité du nombre de clients 'n' :

$\forall$n impair et n>1, alors $\theta_{MAX}(n,2)=S(n,2)$,
$\forall$n pair et n>1, alors $\theta_{MAX}(n,2)>S(n,2)$. *(20)*

Dans le cas général nous avons donc :

$$\forall n \text{ et } \forall k, \quad \theta_{MAX}(n,k) \geq S(n,k), \qquad (21)$$

et par conséquent en appliquant *(17)* :

$$\sum_{k=1}^{n} \theta_{MAX}(n,k) \geq \varpi(n) = \sum_{k=1}^{n} S(n,k)) \qquad (22)$$

On peut le vérifier pour n=6 :
$\varpi(6)=1+31+90+65+15+1=203$,
$\theta_{MAX}(6,1)=6!/6!=1$,
$\theta_{MAX}(6,2)=\{(5,1),(4,2),(3,3)\}=41$,
$\theta_{MAX}(6,3)=\{(4,1,1),(3,1,2),(2,2,2)\}=180$,
$\theta_{MAX}(6,4)=\{(3,1,1,1),(2,2,1,1)\}=130$,
$\theta_{MAX}(6,5)=\{(2,1,1,1,1)\}=360$,
$\theta_{MAX}(6,6)=\{(1,1,1,1,1,1)\}=720$ (toutes redondantes),
$\Sigma_{k=1}^{n} \theta_{MAX}(6,k)=1432>>203$.

III-A5) Les *k-partitions* «utiles» $\psi(n,k)$

Si l'on définit une topologie spécifique, par exemple le «cercle» (cf. la figure *Fig-III-A5-14*), pour positionner les clients (ici cinq) alors on peut introduire la notion de configuration sous-optimale.
Résoudre $SYS_{n,k}$ c'est trouver la solution (sol_p) optimale à $SYS_{n,k}$ ($SYS_{5,2}$ sur la figure *Fig-III-A5-14*). C'est celle qui minimise $CG_{n,k}$, on l'appelle $CG^*_{n,k}$.

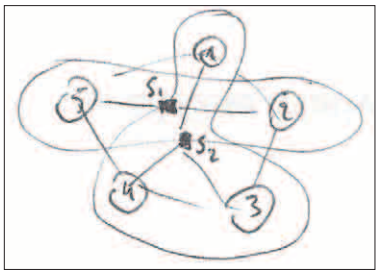


Fig-III-A5-14

Sur l'exemple de la figure *Fig-III-A5-14* la solution sol_1 proposée partitionne $SYS_{5,2}$ de la façon suivante :
$conf_1=\{Part_1,Part_2\}=\{(2,5),(1,3,4)\}$.
Cela conduit à définir $CG(sol_1)$. Les enveloppes convexes des clients de chaque partition $Part_1$ et $Part_2$ ont une intersection non vide. Intuitivement on imagine que cette solution n'est pas optimale. Il suffit de répartir les clients dans les deux partitions afin que les enveloppes convexes résultantes aient une intersection vide. Par exemple pour la solution sol_2 :
$conf_2=\{Part_1,Part_2\}=\{(1,5),(2,3,4)\}$.

En effet chaque serveur S_1 et S_2 sera plus proche de ses clients respectifs et donc nous aurons $CG(sol_2)<CG(sol_1)$.

Toutes les solutions (sol_p) qui ne sont pas optimales ($CG^*_{5,2}<CG(sol_p)$) ne sont pas nécessairement sous-optimales. L'idée est de trouver une propriété liée à la topologie qui permette de disqualifier certaines solutions que nous appellerons sous-optimales. Dans le cas du cercle l'intersection non vide des enveloppes convexes des points de chaque partition peut être prise pour définir les partitions sous-optimales.

Comme nous venons de le voir sur l'exemple de la figure *FIG-III-A5-14*, la notion de *k-partition* «utile» est directement liée à la contextualisation du partitionnement, c.-à-d. son incarnation dans une problématique spécifique, liée à $SYS_{n,k}$.
Nous avons vu que S(n,k) correspond au nombre exact (sans répétition, sans ordre et sans redondance) de *k-partitions* sur un ensemble de 'n' éléments. Ce nombre a une borne supérieure ($\theta_{MAX}(n,k)$) qui inclut les redondances. Malheureusement S(n,k) est très grand et rend incalculable la recherche d'une solution optimale (CG^*) à $SYS_{n,k}$, puisqu'il faut combiner S(n,k) avec le nombre de *k-positions* sur la grille *(7b)*. Dans ce contexte, peut-on minimiser (borne inférieure) le nombre de *k-partitions*?

En positionnant les clients dans une topologie, conformément à la définition de $SYS_{n,k}$, comme ici le cercle, la notion de borne inférieure à S(n,k) peut être introduite.

Définition: On appelle borne inférieure de S(n,k) pour le système $SYS_{n,k}$, que l'on nomme $\underline{\theta}(n,k)$, le nombre des configurations de $SYS_{n,k}$ qui ne sont pas sous-optimales. On appellera ces configurations des *k-partitions* «utile». Nous noterons $\psi(n,k)$ l'ensemble de ces configurations/*k-partitions* «utiles».

$\underline{\theta}(n,k)$

$\psi(n,k)$

Nous obtenons ainsi:
$\theta_{MAX}(n,k) \geq S(n,k) > \underline{\theta}(n,k) = Card(\psi(n,k))$, *(23)*
avec $\psi(n,k) \subset Conf_{q(n,k)}$.

Propriété: Dans le cas général ($\forall n$ et $\forall k$) de la résolution de $SYS_{n,k}$, il existe toujours une borne inférieure $\underline{\theta}(n,k)$ à S(n,k) qui dépend des propriétés de la topologie dans laquelle évolue $SYS_{n,k}$. Nous pouvons formuler cela dans l'autre sens, c.-à-d. en énonçant qu'il existe toujours au moins une topologie qui contraint $SYS_{n,k}$ de telle façon que S(n,k) possède une borne inférieure $\underline{\theta}(n,k)$. Cette borne inférieure est le nombre d'éléments (Card) de $\psi(n,k)$.

Nous avons défini la notion de *k-partition* «utile» à partir de l'intersection topologique de ses partitions, prises deux à deux. Il nous faut préciser cette notion d'intersection topologique vide ou non vide.

Définition: Soit $conf_1$ une *k-partition* de $SYS_{n,k}$, nous avons $conf_1=\{Part_1, Part_2, \ldots, Part_k\}_1$. On définit $\Omega(Part_i, Part_j)$ l'intersection topologique entre les deux partitions $Part_i$ et $Part_j$ de $conf_1$, comme étant le résultat de l'intersection (dans l'espace euclidien) des enveloppes convexes respectives des éléments (clients) de chaque partition $Part_i$ et $Part_j$ de $conf_1$.

$\Omega(Part_i, Part_j)$

intersection topologique

Définition: Une *k-partition* $conf_1=\{Part_1, Part_2, \ldots, Part_k\}$ est dite «utile», c.-à-d. que:
$conf_1 \in \psi(n,k) \Leftrightarrow \Omega(Part_i, Part_j)=\{\emptyset\}$, $\forall i,j \in [1,k]$,
avec $Part_i \subset conf_1$ et $Part_j \subset conf_1$.

Sur la figure *Fig-III-A5-16* nous avons chacune des vingt-et-une *k-partitions* «utiles» de $Conf_{q(7,2)}$.

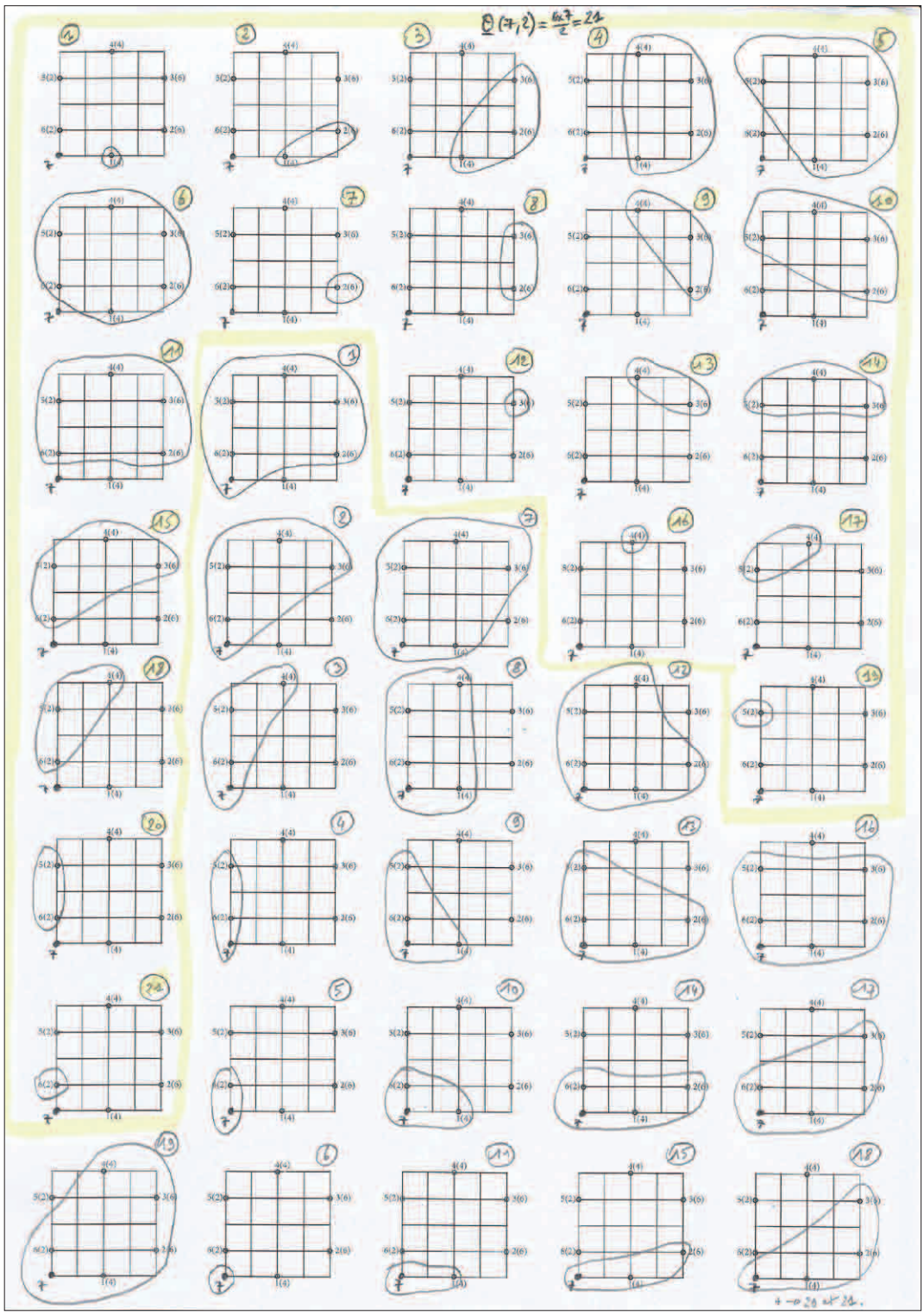


Fig-III-A5-16

Si l'on prend l'exemple $SYS_{6,2}$, vu précédemment, nous avons :

$conf_q(6,2)=\{C1\}\cup\{C2\}\cup\{C3\}$ avec

$C1=Conf_{1,5}$, $C2=Conf_{2,4}$ et $C3=Conf_{3,3}$,

$\theta_{MAX}(6,2)=\theta_{MAX}(6,2)_{C1}+\theta_{MAX}(6,2)_{C2}+\theta_{MAX}(6,2)_{C3}$,

$\theta_{MAX}(6,2)=(6!/1!5!)+(6!/2!4!)+(6!/3!3!)=41$.

On obtient trente-et-une configurations avec S(6,2). En observant les figures *Fig-III-A3-9/10/11* on peut dénombrer les partitions sous-optimales (0 pour C1, 9 pour C2 et 7 pour C3) suivant le critère de l'intersection non vide de l'enveloppe convexe. Nous obtenons :

$\underline{\theta}(6,2)=(6-0)+(15-9)+(10-7)=15$.

On obtient quarante-et-une configurations pour $\theta_{MAX}(6,2)$ contre trente-et-une pour S(6,2). La différence s'explique par les redondances (en grisée sur la figure *FIG-III-A3-11*, uniquement dans C3). Il existe seize configurations sous-optimales, d'où :

$\underline{\theta}(6,2)=15 < S(6,2)=31 < \theta_{MAX}(6,2)=41$.

Cette notion d'intersection topologique repose sur l'idée de minimiser CG *(2)* afin de trouver la configuration dont la valeur CG est optimale (CG^*).

Dans l'espace euclidien nous savons que le point de Fermat 'I' (appelé également la médiane géométrique) d'un triangle (ABC) est le seul point qui minimise la somme des distances des trois sommets 'A', 'B' et 'C' à un point (IA+IB+IC). De même le problème du cercle minimum (également valable pour une sphère minimale), également appelé le problème de Fermat-Torricelli-Steiner, cherche à trouver le point qui minimise la somme de sa distance à un ensemble de 'p' points du plan $(\inf_x \max_{1\le i\le p}||x-ai||)^2$.
Le problème appelé ESFL généralise celui de Fermat-Torricelli-Steiner en ajoutant une pondération sur chaque distance $(\inf_x \max_{1\le i\le p} w_i||x-ai||)$. On montre qu'un tel point correspond au centre du cercle minimal qui inclut les 'p' points considérés. Ce point est donc inclus dans l'enveloppe convexe de ces 'p' points, puisqu'elle est englobée par le cercle minimal.
Nous pouvons vérifier cette notion d'utilité en calculant les valeur CG des différentes configurations rencontrées dans nos exemples.

Intuitivement, si nous prenons la configuration $conf_1=\{Part_1,Part_2\}_1$ (k=2), telle que $\Omega(Part_1,Part_2)\neq\{\emptyset\}$ cela signifie que l'intersection des enveloppes convexes de $Part_1$ et de $Part_2$ n'est pas nulle.
On peut montrer simplement que dans ce cas l'enveloppe convexe de chacune des deux partitions est égale au cercle sur lequel figurent les 'n' clients de $SYS_{n,k}$. Ce cercle correspond au cercle minimal englobant les clients des différentes partitions. Si 'α' est le nombre de clients de $Part_1$ et 'β' celui de $Part_2$, avec α+β=n, alors d'après *(2)* :

$CG(sol_p)=cg(sol_p,S_1)+cg(sol_p,S_2)=\alpha R+\beta R=nR$.

où 'R' est le rayon du cercle minimal, ici celui des clients.

2 L'algorithme de Weiszfeld résoud ce problème en utilisant une technique de *climbing*. Il est simple et converge rapidement : E. Weiszfeld, «*Sur le point pour lequel la somme des distances de 'n' points donnés est minimum*», Tohoku Mathematical Journal, vol. 43, 1937, p.355-386.

Or, d'après Weiszfeld, nxR représente la valeur optimale pour k=1 (un seul serveur), c.-à-d. le cercle minimal qui englobe tous les 'n' points, tous positionnés sur le cercle. En déplaçant l'un des serveurs sur une position différente du centre du cercle de départ, tout en restant dans ce cercle, la partition correspondante aura un cercle minimal plus petit et le CG correspondant sera inférieur au précédent. On voit que nxR est une borne supérieure des valeurs optimales de CG pour sol_p.

Ce point corrobore la loi déjà évoquée qui consiste à définir $CG^*_{n,k}$ comme étant une fonction décroissante en fonction de 'k', avec $k\epsilon[1,n]$. Plus 'k' tend vers 'n' est plus $CG^*_{n,k}$ diminue pour valoir 0 si k=n :

$CG^*_{n,k} > CG^*_{n,k+1}$ $\forall k\epsilon[1,n]$ (cf. la figure *Fig-III-A3-8*).

Nous nous appuierons sur cette propriété pour aller plus loin dans la définition et le calcul des *k-partitions* «utiles», $\forall k\epsilon[1,n]$ en appliquant une deuxième contrainte/ filtre.

k-partition «utile»

Définition: Une *k-partition* est dite «utile» si elle est à intersection topologique nulle, et si elle vérifie $CG_{n,k} < CG^*_{n,k-1}$ (principe de la *descente du gradient d'optimalité, DGO*).

Autrement dit, on ne retient pas dans $\psi(n,k)$ les *k-partitions* à intersection topologique nulle dont le CG est supérieur au CG optimal avec un serveur de moins.
De tels cas existent. On peut le voir sur les exemples des figures *Fig-III-A1-2* pour S(6,1), *Fig-III-A3-9/10/11* pour S(6,2) et *Fig-III-A3-12b/12c/12d* pour S(6,3).
Nous avons :

a) $CG(sol_p)\epsilon[64,112]$, avec $sol_p \subset Conf_{q(6,1)}$, $S(6,1) = \underline{\theta}(6,1) = 1$ et $CG^*_{6,1} = 64$;
b) $CG(sol_p)\epsilon[38,64]$, avec $sol_p \subset Conf_{q(6,2)}$, $S(6,2) = 31$, $\underline{\theta}(6,2) = 15$ et $CG^*_{6,2} = 38$;
c) $CG(sol_p)\epsilon[24,44]$, avec $sol_p \subset Conf_{q(6,3)}$, $S(6,3) = 90$, $\underline{\theta}(6,3) = 50$ et $CG^*_{6,3} = 24$.

Dans le cas des *2-partitions* (b) nous pouvons observer en jaune sur la figure *Fig-III-A5-17* les *2-partitions* «utiles» au sens «intersection topologique vide»: a-1, a-2, a-3, a-4, a-5, a-6, b-1, b-2, b-3, b-4, b-5, b-6, c-1, c-2 et c-3. On trouve parmi ces partitions utiles les partitions optimales.
On peut remarquer que certaines partitions «utiles» sont loin d'être optimales. C'est le cas en particulier pour a-1 (CG=64), a-5 (CG=64) et a-6 (CG=64). En appliquant la nouvelle définition des partitions «utiles» on voit que ces trois dernières partitions (a-1, a-5 et a-6) ont $CG=CG^*_{6,1}$, ce qui n'est pas conforme à la nouvelle définition. Elles ne seront pas conservées.
Sur notre exemple (cf. la figure *Fig-III-A5-17)* on peut dire que $\underline{\theta}(6,2)=12$ et non 15. Cela montre bien que *(24)* est une borne supérieure du nombre de *2-partitions* «utiles».
De même nous avons trois configurations de $\psi(6,3)$ dont CG est supérieur ou égal à $CG^*_{6,2}=38$ (cf. les configurations entourées de jaune sur la figure *Fig-III-A5-18*). On en déduit que $\underline{\theta}(6,3)=47$.

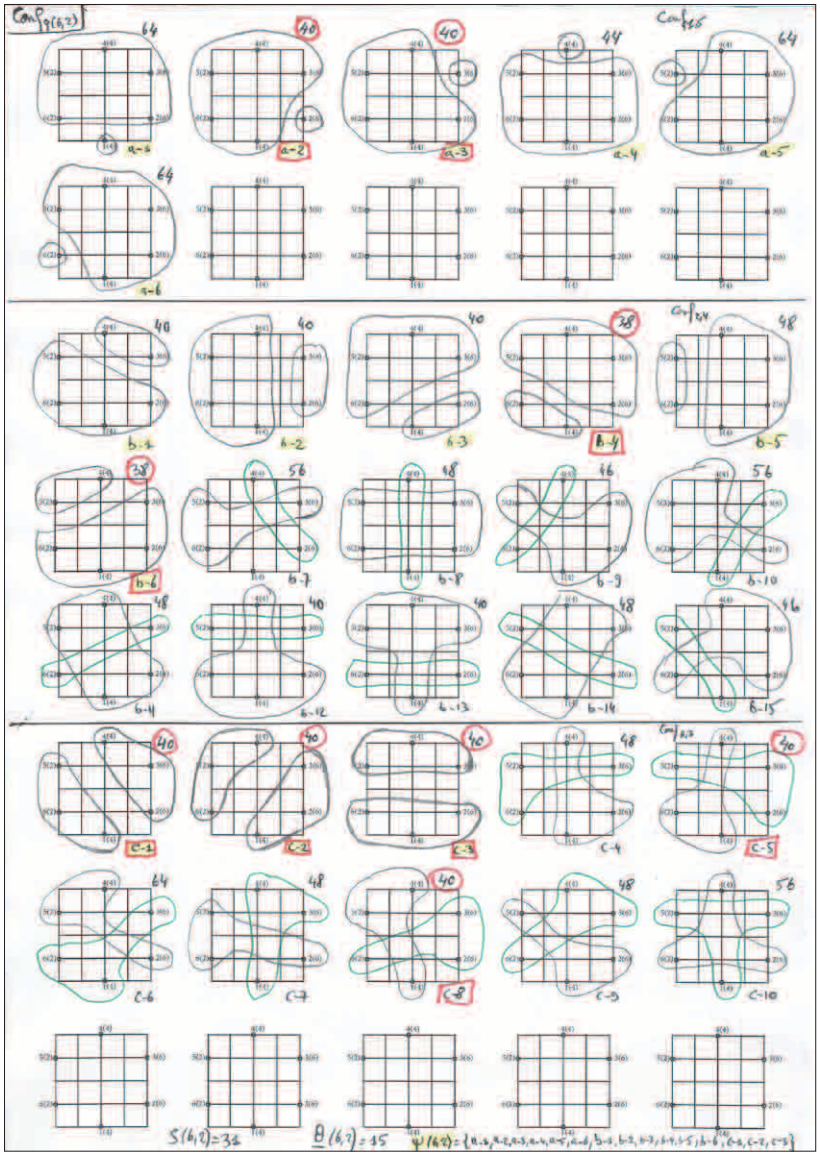


Fig-III-A5-17

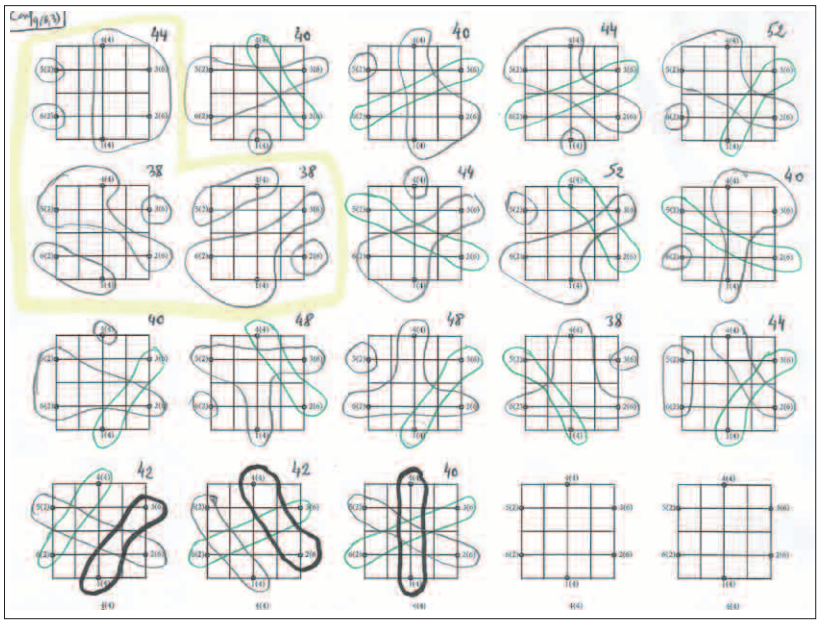


Fig-III-A5-18

La figure *Fig-III-A5-18* nous montre également l'ensemble des dix-huit configurations sur les quatre-vingt-dix *3-partitions* (S(6,3)) pour lesquelles leur CG est supérieur ou égal à $CG^{*}_{6,2}$.

CALCUL DES *2-PARTITIONS* «UTILES» : $\underline{\theta}$(N,2)

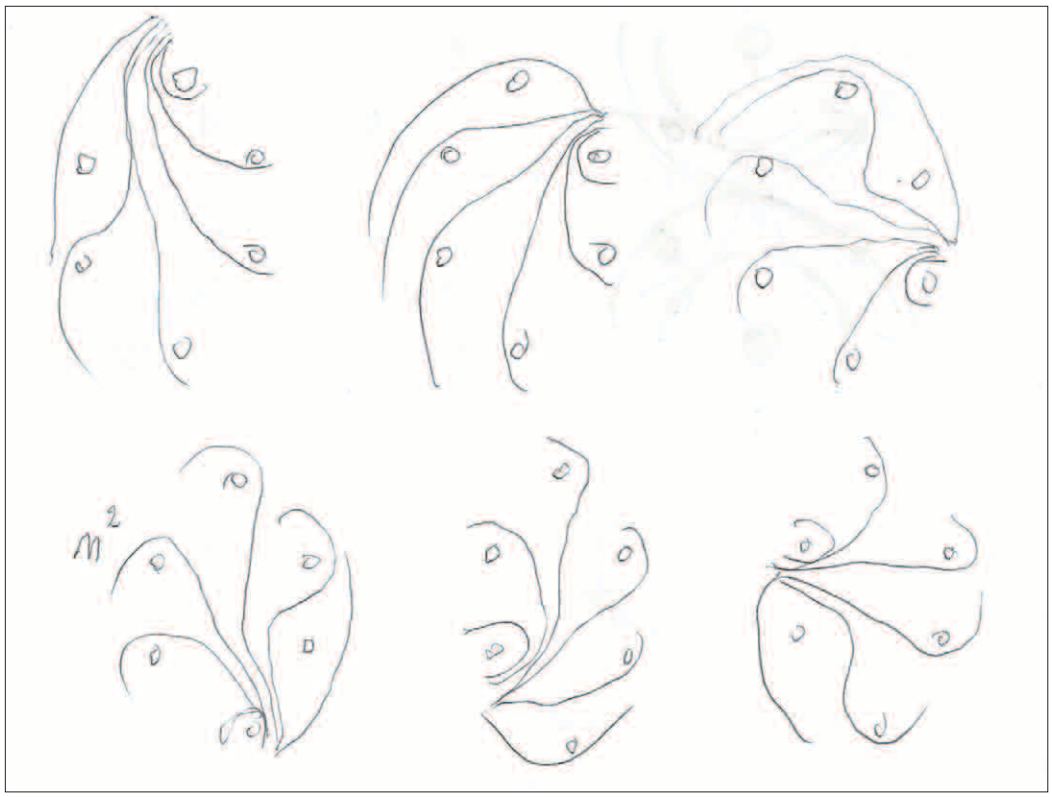


Fig-III-A5-19

Pour illustrer cette propriété, prenons comme topologie le cercle (sur la figure *Fig-III-A5-19*) avec deux serveurs (k=2) et regardons si l'on peut calculer le nombre de configurations utiles $\underline{\theta}(n,2)$, $\forall n$.

Cette topologie est intéressante car elle peut être assimilée à un capteur en périphérie d'une cellule d'encodage de connaissances, à l'image des membranes cellulaires. Nous reviendrons sur cette proposition dans les chapitres suivants en particulier quand nous ferons référence au concept d'autopoïèse proposé par Maturana et Varela. Nous pourrions aussi prendre une sphère ou d'autres topologies plus complexes.

Soit C_nS_2 ('n' clients et k=2 serveurs) où l'on ne regarde que les partitions «utiles», c.-à-d. dont l'intersection topologique des partitions deux à deux est vide. Comme l'indique la figure *Fig-III-A5-19* chaque client (rond sur le cercle) peut construire (n-1) configurations, ce qui donne $\underline{\theta}(n,2)=n(n-1)$. Les configurations obtenues étant symétriques il nous faut diviser cette valeur par deux:

$$\underline{\theta}(n,2)=n(n-1)/2. \qquad (24)$$

Si l'on veut tenir compte du cas particulier où l'on a 'n' clients dans une partition et zéro dans l'autre, il faut ajouter 1 à la formule. Nous obtenons ainsi :

$$\underline{\theta}(n,2)=(n(n-1)/2)+1. \qquad (24b)$$

En réalité ces formules *(24 et 24b)* nous donnent une borne supérieure du nombre de *k-partitions* «utiles». En effet nous pouvons combiner d'autres filtres pour diminuer cette valeur comme celui décrit précédemment qui est basé sur CG^{*}_{k-1}.

Pour les situations suivantes (variation du nombre de clients) nous obtenons :

- n=6→ $\underline{\theta}(6,2)=15$, $S(6,2)=31$;
- n=7→ $\underline{\theta}(7,2)=21$, $S(7,2)=63$ (cf. la figure *Fig-III-A5-16*) ;
- n=21→ $\underline{\theta}(21,2)=210$, $S(21,2)=1\ 048\ 575$.

Le nombre des *k-partitions* pour un ensemble de 'n' clients est minimisé par $\underline{\theta}(n,k)$ en fonction des propriétés de la topologie de $SYS_{n,k}$. Par contre il ne dépend directement ni de la dimension de la grille, ni de son maillage dans cette topologie. En revanche nous avons vu *(7b)* que le nombre de solutions sol_p ($card(SOL_{n,k})$) dépend du produit des *k-partitions* par le nombre ($Card(POS_k)$) des *k-positions* (positionnées sur la grille).
Si nous avons n=21, quelque soit la grille, avec k=2 nous aurons deux-cent-dix *2-partitions* «utiles». En revanche, dans ce cas, si nous prenons une grille de cent-quatre-vingt-seize (14x14) unités nous obtenons :

$\underline{\theta}(21,2)=210$, *(24)*
$S(21,2)=1\ 048\ 575$, *(12)*
$Card(POS_2)=19110$, *(7)*
$Card(SOL_{21,2})=4\ 013\ 100$, *(7) et (24)*
$Card(SOL_{21,2})=20\ 038\ 268\ 250$. *(7) et (12)*

avec une grille de quatre cents (20x20) unités cela donne :

$\underline{\theta}(21,2)=210$, *(24)*
$S(21,2)=1\ 048\ 575$, *(12)*
$Card(POS_2)=79800$, *(7)*
$Card(SOL_{21,2})=16\ 758\ 000$, *(7) et (24)*
$Card(SOL_{21,2})=83\ 676\ 285\ 000$. *(7) et (12)*

Nous voyons clairement l'impact des caractéristiques de la grille (dimension et maillage) sur le nombre de solutions parmi lesquelles se cachent les solutions optimales que l'on cherche. Nous reviendrons sur cette notion pour affiner nos heuristiques.

Pour trouver les solutions optimales et étudier leurs propriétés, comme nous allons le faire dans la suite de cet exposé, nous appliquerons le filtre de la deuxième condition ($CG_{n,k}<CG^{*}_{n,k-1}$) pour déterminer les *k-partitions* «utiles», en comparant leur CG avec $CG^{*}_{n,k-1}$. Le calcul sera fait de façon itérative, c.-à-d. en commençant avec k=1 pour obtenir $CG^{*}_{n,1}$, puis k=2,…, jusqu'à k=n.
En attendant cette étude qui nécessite de construire des outils adhoc, nous allons poursuivre notre étude des propriétés de $\underline{\theta}(n,k)$, la borne inférieure de $S(n,k)$, qui dépend de la topologie de $SYS_{n,k}$.

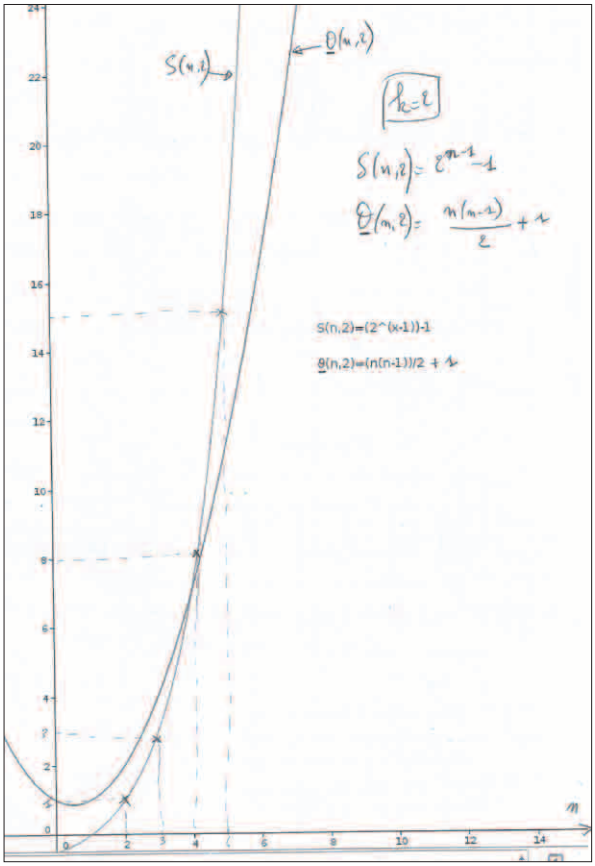


Fig-III-A5-19b

Nous avons exploré tous les cas ($\forall n>1$) de $\underline{\theta}(n,2)$ pour les 'n' clients placés en *2-partition* sur un cercle (formule *(24)*). Peut-on trouver une formule générale, en 'k' et en 'n', du calcul de $\underline{\theta}(n,k)$ pour une topologie donnée?

Dans la mesure où nous avons une formule générale pour k=2 avec les 'n' clients sur un cercle (qui pourrait être étendue à une sphère), et que cette topologie nous intéresse particulièrement (cf. l'orientation autopoïétique des heuristiques proposées dans les chapitres suivants), nous allons étudier $\underline{\theta}(n,k)$, avec les 'n' clients sur le cercle, mais cette fois pour des valeurs de k>2.

RÉSUMÉ DES DIFFÉRENTS CALCULS DES CONFIGURATIONS DE $SYS_{6,2}$

$Conf_{q(n,p)}$ avec n=6 et p=2	$Conf_{1,5}$	$Conf_{2,6}$	$Conf_{3,3}$	total	commentaires
$A_n^p=n^p$	6	36	216	258	Arrangements avec répétition (ii), ordonnés (ij≠ji) et avec doublon {(i,j),(i,j)}.
$A_n^p=$ n!/(n-p)!	6	30	120	156	Arrangements sans répétition, ordonnés et avec doublon.
$A_n^p=$ n!/p!(n-p)!	6	15	20	41	Arrangements sans répétition, sans ordre (ij⇔ji) et avec doublon.
$\theta_{MAX}(6,2)$	6	15	20	41	Borne supérieure de S(n,2) avec $S(n,2)=\theta_{MAX}(n,2)$ si 'n' est impair et $S(n,2)<\theta_{MAX}(n,2)$ si 'n' est pair.
S(6,2)	-	-	-	31	*2-partition* de N à 'n' éléments, sans répétition, sans ordre et sans doublon.
$\underline{\theta}(6,2)$	6	6	3	15	*2-partition* «utile».
$\varpi(6)_{p=1,6}$	-	-	-	203	Somme des S(n,p) avec p∈[1,n], sans répétition, sans ordre et sans doublon.

CALCUL DES *3-PARTITIONS* «UTILES» : $\underline{\theta}(N,3)$

Pour dénombrer les *3-partitions* «utiles» sur le cercle, nous allons partir de $\underline{\theta}(6,3)$, avec six clients, regarder toutes les situations (cf. les figures *Fig-III-A5-20/21/22/23*), pour essayer de comprendre les lois sous-jacentes à la topologie et en déduire des formules pour des valeurs de 'k' supérieures.

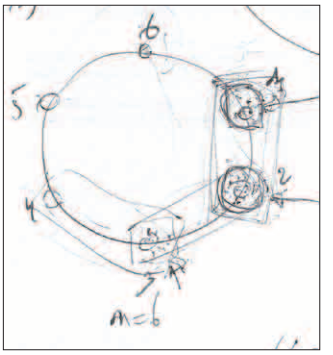


Fig-III-A5-24

L'exemple $SYS_{6,3}$ est décrit par:

$Conf_q(6,3)=\{C1\}\cup\{C2\}\cup\{C3\}$ avec

$C1=Conf_{1,1,4}$, $C2=Conf_{1,2,3}$ et $C3=Conf_{2,2,2}$.

Le nombre des configurations «utiles» se décompose suivant C1, C2 et C3 de la façon suivante:

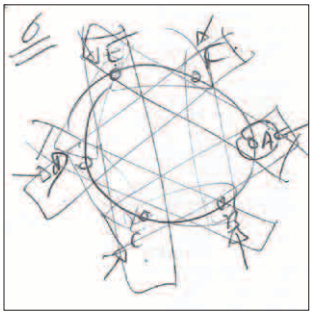


Fig-III-A5-25

(1,1,4)	+	(1,2,3)	+	(2,2,2)	=	avec n=6	total
(n-1)		(n-1)		1		2n-1	11
(n-1)-1		(n-1)		1		2n-2	10
…		…		…		…	…
(n-1)-(n-2)		(n-1)		1		n+1	7
(n-1)-(n-1)		(n-1)		0		n-1	5
n(n-1)/2		**n(n-1)**		**(n-1)**		**[n/2(3n-1)]-1**	**50**
15		**30**		**5**		**50**	**-**

Le nombre d'éléments de C1 est une suite arithmétique $(U_p+U_{p+1}+\ldots+U_m)$ avec $U_p=0$ et $U_m=(n-1)$ dont la somme vaut:

$U_p+U_{p+1}+\ldots+U_m=(m-p+1)\times(U_0+U_m)/2=n(n-1)/2$,

avec $m=n-1$ et $p=0$.

$\underline{\theta}(6,3)$ est une suite arithmétique, avec $U_p=n$ et $U_m=2n-1$ dans laquelle il faut gérer le décalage de 1 dû à C3:

$\underline{\theta}(6,3)=(U_p+U_{p+1}+\ldots+U_m)-1=[(m-p+1)\times(U_0+U_m)/2]-1$

$\underline{\theta}(6,3)=[n/2(3n-1)]-1=51-1=50$.

Si l'on passe à $SYS_{7,3}$ nous avons les configurations suivantes:

$Conf_q(7,3)=\{C1\}\cup\{C2\}\cup\{C3\}\cup\{C4\}$ avec

$C1=Conf_{1,1,5}$, $C2=Conf_{1,2,4}$, $C3=Conf_{1,3,3}$ et $C4=Conf_{2,2,3}$.

(1,1,5)	+	(1,2,4)	+	(1,3,3)	+	(2,2,3)	=	avec n=7	total
(n-1)		(n-2)+1		(n-4)		[(n-3)/2]+1		1/2(7n-13)	18
(n-1)-1		(n-2)+1		(n-4)		[(n-3)/2]+1		1/2(7n-13)-1	17
…		…		…		…		…	…
(n-1)-(n-2)		(n-2)+1		(n-4)		[(n-3)/2]+1		1/2(7n-13)-(n-2)	13
(n-1)-(n-1)		(n-2)+1		(n-4)		[(n-3)/2]+1		1/2(7n-13)-(n-1)	12
n(n-1)/2		**n(n-1)**		**n(n-4)**		**n(n-1)/2**		**3n(n-2)**	**105**
21		**42**		**21**		**21**		**105**	**-**

Pour le calcul de $\underline{\theta}(7,3)$ nous avons encore à faire à une suite arithmétique avec $U_m=1/2(7n+13)$ et $U_p=1/2(5n-11)$.

$\underline{\theta}(7,3)=U_p+U_{p+1}+\ldots+U_m=3n(n-2)=3x7x5=105$

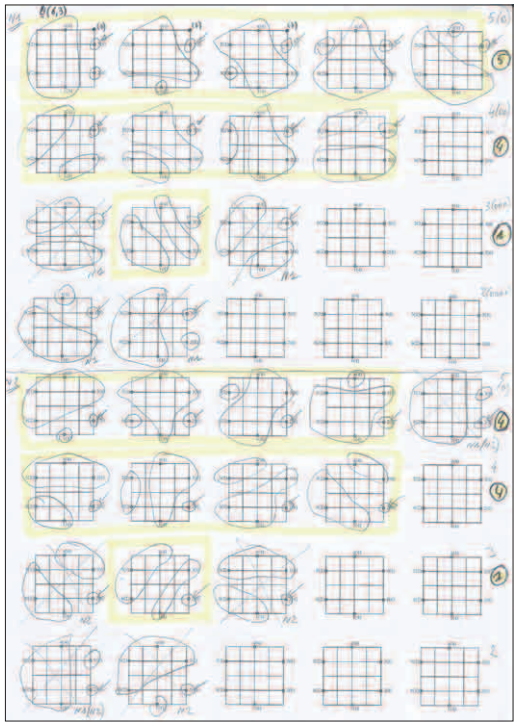


Fig-III-A5-20

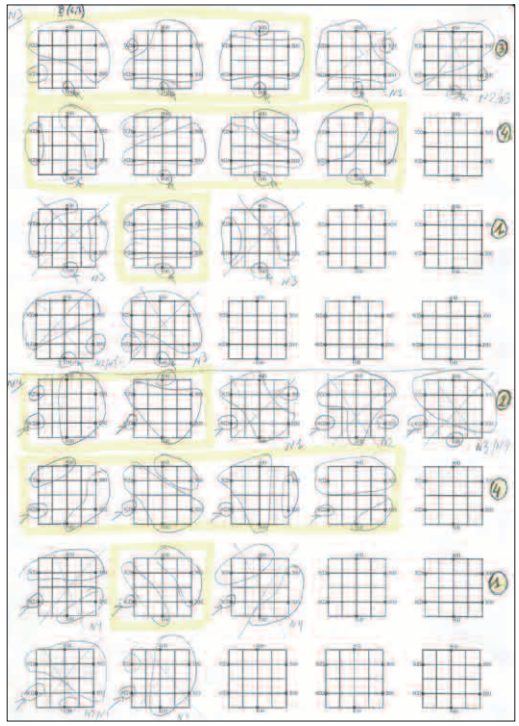


Fig-III-A5-21

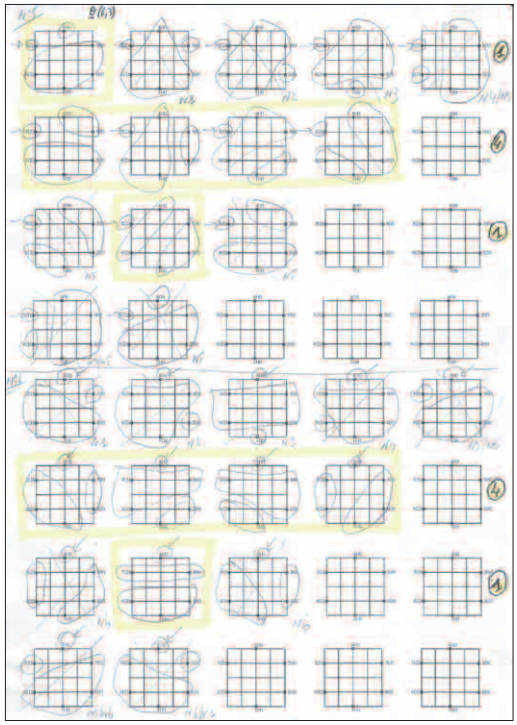


Fig-III-A5-22

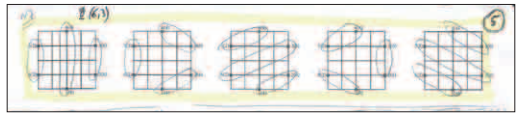


Fig-III-A5-23

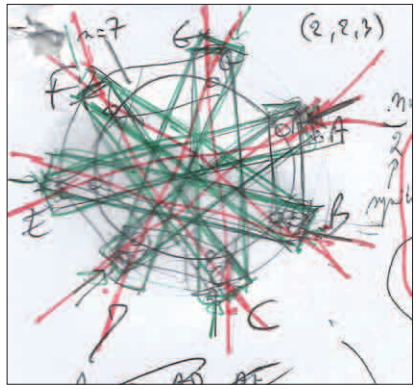


Fig-III-A5-26

Poursuivons avec $SYS_{8,3}$, nous avons les configurations suivantes :

$Conf_q(8,3)=\{C1\}\cup\{C2\}\cup\{C3\}\cup\{C4\}\cup\{C5\}$ avec
$C1=Conf_{1,1,6}$, $C2=Conf_{1,2,5}$, $C3=Conf_{1,3,4}$, $C4=Conf_{2,2,4}$ et $C5=Conf_{2,3,3}$.

(1,1,6)	+	(1,2,5)	+	(1,3,4)	+	(2,2,4)	+	(2,3,3)	=	avec n=8	total
(n-1)		(n-2)+1		(n-1)		[(n-3)/2]+1		(n-5)+1/2		1/2(9n-16)	28
(n-1)-1		(n-2)+1		(n-1)		[(n-3)/2]+1		(n-5)+1/2		1/2(9n-16)-1	27
…		…		…		…		…		…	…
(n-1)-(n-2)		(n-2)+1		(n-1)		[(n-3)/2]+1		(n-5)+1/2		1/2(9n-16)-(n-2)	22
(n-1)-(n-1)		(n-2)+1		(n-1)		[(n-3)/2]+1		(n-5)+1/2		1/2(9n-16)-(n-1)	21
n(n-1)/2		**n(n-1)**		**n(n-1)**		**n(n-1)/2**		**n(n-9/2)**		**n/2(8n-15)**	**196**
28		**56**		**56**		**28**		**28**		**196**	**-**

Pour le calcul de $\underline{\theta}(8,3)$ nous avons encore à faire à une suite arithmétique avec $U_m=9/2n-8$ et $U_p=7(n/2-1)$:

$\underline{\theta}(8,3)=U_p+U_{p+1}+\ldots+U_m=n/2(8n-15)=4x51=105$.

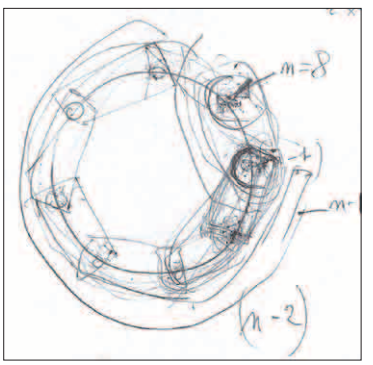


Fig-III-A5-27

Nous voyons apparaître des règles liées aux symétries et aux contraintes du cercle. Les singletons (partition à un seul élément) jouent un rôle important, de même que les doublons (deux partitions distinctes possédant le même nombre d'éléments).

Au vu de ces premiers résultats nous pouvons définir avec certitude, mais sans l'avoir expérimenté exhaustivement, comme pour $\underline{\theta}(6,3)$, $\underline{\theta}(7,3)$ et $\underline{\theta}(8,3)$, les formules pour $\underline{\theta}(9,3)$.
D'autres développements mathématiques seraient nécessaire pour généraliser ces formules pour des valeurs de 'n' supérieures. Nous n'en n'aurons pas besoin ici, mais nous voyons le principe et proposerons la courbe qui extrapole ces calculs.

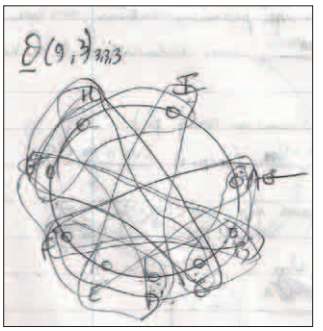


Fig-III-A5-29

Les formules pour $SYS_{9,3}$, suivent les configurations suivantes et les propriétés topologiques vues précédemment :

$Conf_q(9,3)=\{C1\}\cup\{C2\}\cup\{C3\}\cup\{C4\}\cup\{C5\}\cup\{C6\}\cup\{C7\}\cup\{C8\}$ avec
$C1=Conf_{1,1,7}$, $C2=Conf_{1,2,6}$, $C3=Conf_{1,3,5}$, $C4=Conf_{1,4,4}$, $C5=Conf_{2,2,5}$, $C6=Conf_{2,3,4}$, $C7=Conf_{2,4,4}$ et $C8=Conf_{3,3,3}$.

(1,1,7)	+	(1,2,6)	+	(1,3,5)	+	(1,4,4)	+	(2,2,5)	+	(2,3,4)	+	(2,4,4)	+	(3,3,3)	=	n=9	total
n(n-1)/2		**n(n-1)**		**n(n-1)**		**n(n-5)**		**n(n-1)/2**		**n(n-1)**		**n(n-1)/2**		**n+3**		**n/2(11n-15)-6**	**372**
36		**72**		**72**		**36**		**36**		**72**		**36**		**12**		**372**	**-**

Pour le calcul de $\underline{\theta}(9,3)$ nous avons toujours à faire à une suite arithmétique, mais le nombre de configurations de C8 (n+3) ne tombe pas sur un multiple de 'n' pour n=9. Nous allons donc calculer la suite $(U_p+U_{p+1}+\ldots+U_m)$ sachant qu'elle correspondra à $\underline{\theta}(9,3)$ moins ce qu'il faut pour tomber sur un multiple de 'n'.
Card(C8)=n+3=9+3=12. Nous allons prendre 12+6=18 un multiple de 9 :

$\underline{\theta}(9,3)=(U_p+U_{p+1}+\ldots+U_m)-6$, avec $U_m=6n-8$ et $U_p=5n-7$,

$\underline{\theta}(9,3)=U_p+U_{p+1}+\ldots+U_m=[n/2(11n-15)]-6=378-6=372$.

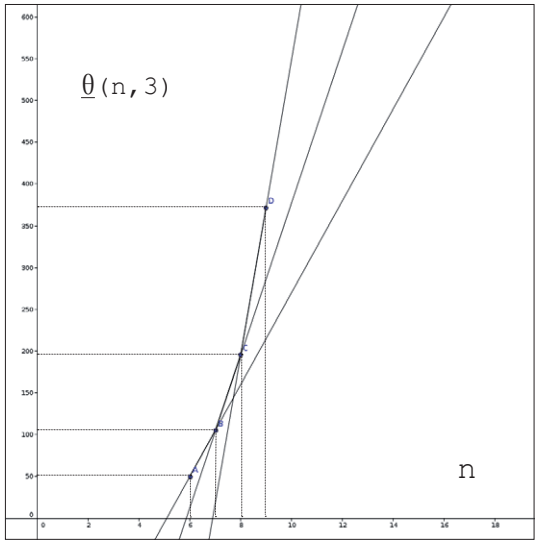


Fig-III-A5-28

La figure *Fig-III-A5-28* nous permet d'extrapoler sur les valeurs de $\underline{\theta}(n,3)$ pour des valeurs quelconques de 'n'.

On peut voir sur la figure *Fig-III-A5-30* les génératrices des valeurs de $\underline{\theta}(n,3)$ en fonction de 'n'. Elles passent toutes par les valeurs correspondantes, calculées précédemment.
La courbe de $\underline{\theta}(n,2)$ a été également placée sur le graphique. Elle est très en-dessous des génératrices de $\underline{\theta}(n,3)$.

Ces résultats représentent une borne supérieure des *k-partitions* «utiles» $\underline{\theta}(n,K)$. Seul le critère d'intersection topologique a été pris en compte. Nous approfondirons, grâce aux explorations informatiques du chapitre suivant, les propriétés des *k-partitions*. *Nous verrons* comment l'application du critère de performance ($CG_{n,k}<CG^{*}_{n,k-1}$) permettra de réduire encore le nombre de ces *k-partitions* «utiles».

k	**2**				**3**			
n	**6**	**7**	**8**	**9**	**6**	**7**	**8**	**9**
$\theta_{MAX}(n,k)$	41	63	162	255	180	497	1474	5469
$S(n,k)$	31	63	127	255	90	301	966	3025
$\underline{\theta}(n,K)$	15	21	28	36	50	105	196	372

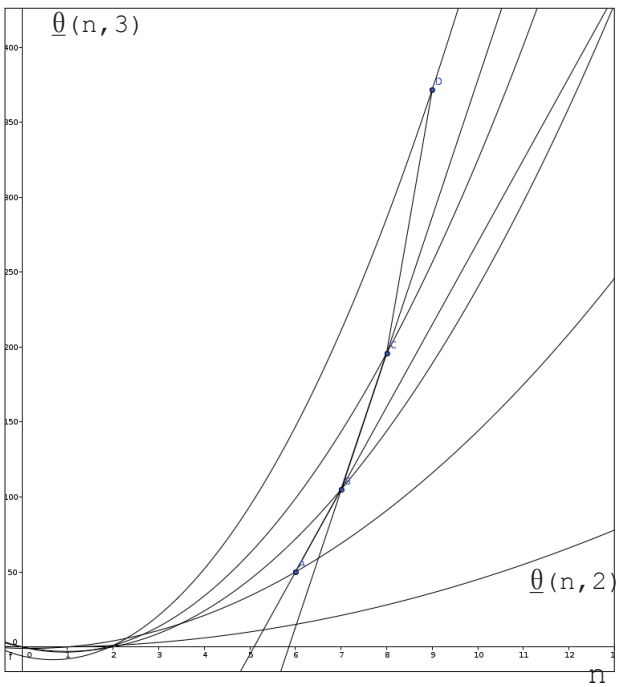


Fig-III-A5-30

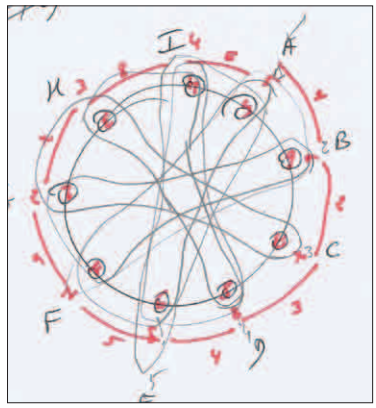


Fig-III-A5-31

ÉTUDE THÉORIQUE ET PREMIERS ENSEIGNEMENTS

III-B

Nous avons vu que le système $SYS_{n,k}$, de type C_nS_k, est un système ouvert, dynamique et décentralisé dans lequel 'n' clients nécessitent simultanément un accès à des ressources au travers de 'k' serveurs (avec k<n) dans un contexte donné.
L'objectif fixé est de trouver la (ou les) solution(s) optimale(s)(sol_p) qui minimise(nt) une grandeur globale ($CG(sol_p)=CG^*_{n,k}$).
Pour cela nous avons dû introduire les notions de grille, de *k-position*, appelée multi-points (MM_{k-i}), et de *k-partition*, appelée configuration ($conf_1$).
Dans la partie III-A nous avons regardé la complexité en temps pour trouver cette solution optimale ($CG^*_{n,k}$). Nous avons trouvé une borne inférieure ($\underline{\theta}(n,k)=Card(\psi(n,k))$) au nombre exact de *k-partition* (S(n,k)) qui nécessite le recours à une topologie (le cercle dans notre exemple). Malheureusement, même pour cette borne inférieure, les valeurs théoriques ne sont pas calculables à moyen et grand horizon (celui d'Internet), c.-à-d. pour de grandes valeurs de 'n' et de 'k'.
Une autre façon de trouver cette borne inférieure ($\psi(n,k)$) passe par l'application du principe de «descente du gradient d'optimalité» (DGO).
Nous allons préciser le rôle de la grille et de la topologie dans la résolution de $SYS_{n,k}$ à travers l'usage de DGO. En réalité la grille peut être vue comme un capteur dont le rôle est d'assurer l'interface entre l'information externe au système et la représentation interne qu'il en fera. Cela va nous conduire à généraliser le problème de type C_nS_k à celui de la représentation des connaissances. Pour clarifier notre propos, nous nous appuierons sur la représentation de l'alphabet latin.

III-B1) Rappels et premières propriétés

Un système $SYS_{n,k}$, de type C_nS_k, est défini par l'ensemble CL, composé de 'n' clients, et l'ensemble SE composé de 'k' serveurs :

$$CL=\{C_1,C_2,\ldots,C_n\}, \text{ avec } n>0,$$
$$SE=\{S_1,S_2,\ldots,S_k\}, \text{ avec } k>0.$$

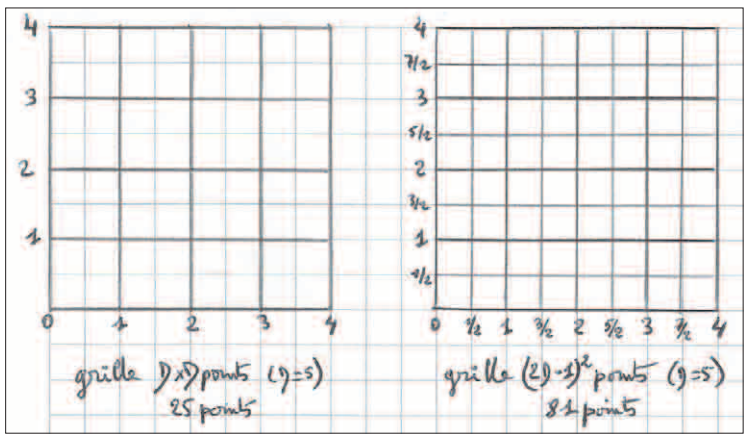


Fig-III-B1-1

$SYS_{n,k}$ est défini dans un espace (graphe, espace euclidien,…) dans lequel on crée une grille. Une grille se caractérise par sa dimension ('D') et la dimension de sa maille.
Sur la figure *Fig-III-B1-1* nous avons deux grilles de dimension D=5. Celle de gauche à une maille d'une unité lui donnant vingt-cinq points, alors que celle de droite à une maille de ½ unité, lui donnant quatre-vingt-un points.

La grille permet de placer les 'n' clients. Ce placement fait partie de la définition du problème $SYS_{n,k}$ à résoudre.

À faire
Nous développerons, dans la partie III-C, des outils capables de paramétrer ces caractéristiques ('D' et maille).

Une fois les clients posés sur la grille il faut y placer des serveurs (*k-position*) pour répondre à leur besoin. Résoudre $SYS_{n,k}$ revient alors à trouver la (ou les) *k-position(s)* qui minimise(nt) CG *(2)*.
Cette approche nécessite pour chacune des *k-partitions* ($conf_1$) possibles (cf. S(n,k)), de calculer la valeur CG de chaque *k-position* (multi-points MM_{k-p}) sur la grille (conformément à la formule *(8)*). Le but est de trouver la solution optimale, celle qui minimise CG.
Les notions de *k-partition* et de *k-position*, bien qu'indépendantes, sont liées entre elles dans cette recherche de la solution optimale.

Pour cela nous pouvons :
- soit fixer une *k-position* (par exemple (S1,S2) sur la partie gauche de la figure *Fig-III-B1-2*) et chercher toutes les *k-partitions* ($conf_1$) possibles (cf. la figure *Fig-III-B1-3*) ;
- soit, à l'inverse, fixer une *k-partition* (par exemple $conf_1=\{Part_1, Part_2\}$ sur la partie droite de la figure *Fig-III-B1-2*) et chercher toutes les *k-positions* possibles (cf. la figure *Fig-III-B1-4*).

Sur notre exemple, les valeurs CG obtenues (de 38 à 70) sont variables.

Pour une *k-partition* donnée, nous pouvons avoir plusieurs *k-positions* différentes qui auront la même valeur de CG. C'est le cas sur la figure *Fig-III-B1-9* pour $CG^*_{6,1}$ (en haut à gauche) où neuf *k-positions* ont CG=64. Nous pouvons avoir aussi, pour une *k-partition* donnée, des valeurs différentes de CG en fonction des *k-positions*.

Nous verrons dans les simulations (au chapitre IV «heuristiques résolutives…») que les serveurs peuvent se déplacer de façon continue dans l'espace euclidien à deux dimensions, tout en gardant l'idée de la grille pour positionner les clients. Dans pareil cas la (ou les) solution(s) ne se mesure(nt) pas en *k-position*, mais en surface. C'est ce que nous pouvons voir sur la figure *Fig-III-B1-5* où les surfaces «solution» apparaissent hachurées en rouge. Ces surfaces se dessinent également sur les figures *Fig-III-B1-6/7* avec un code couleur pour en délimiter les contours.
Nous avons montré (dans la partie III-A) que si les 'n' clients se trouvent sur une topologie particulière sur la grille, par exemple un cercle, alors nous pouvons trouver un sous-ensemble des *k-partitions* ($\psi(n,k)$), dont les membres sont appelés des *k-partitions* «utiles». Il suffit pour cela d'appliquer aux *k-partitions* une propriété liée à la topologie (par exemple l'intersection nulle des enveloppes convexes des partitions $Part_i$ de $conf_1$, dans le cas du cercle).

L'étude menée dans la partie III-A ($\underline{\theta}(n,2)$ pour k=2, ou les droites génératrices avec $\underline{\theta}(n,3)$ pour k=3) montre que dans le cas du cercle, le nombre des *k-partitions* «utiles» ($\underline{\theta}(n,k)$) est très inférieur à S(n,k), mais reste quand même très élevé.

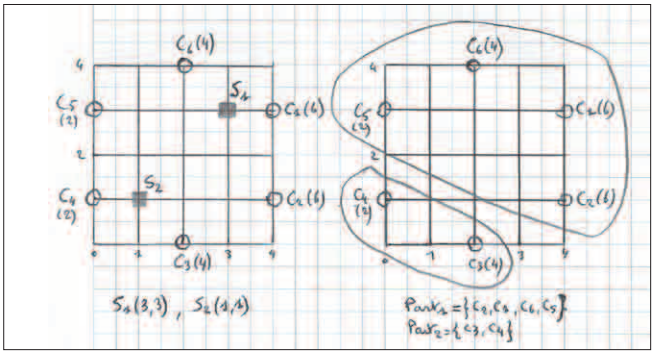


Fig-III-B1-2

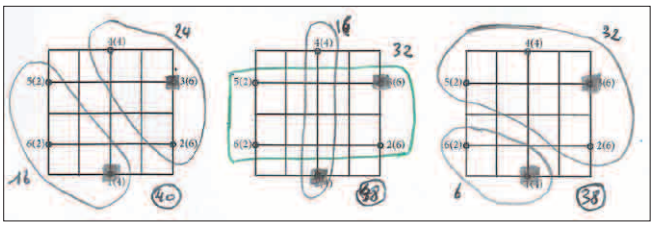


Fig-III-B1-3

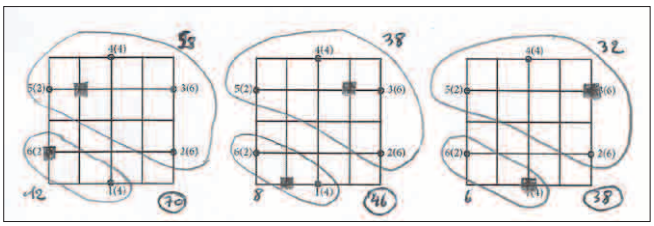


Fig-III-B1-4

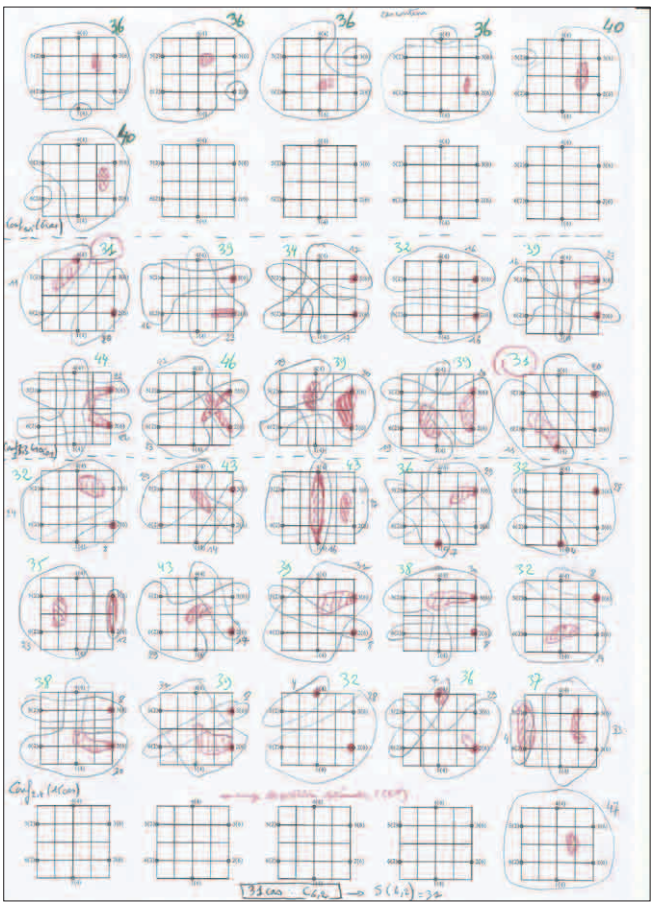


Fig-III-B1-5

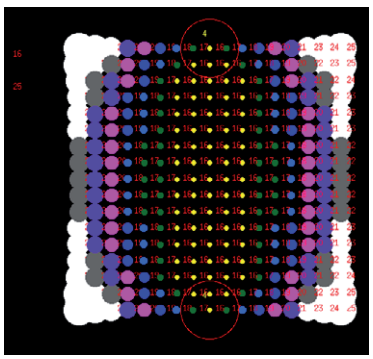


Fig-III-B1-6

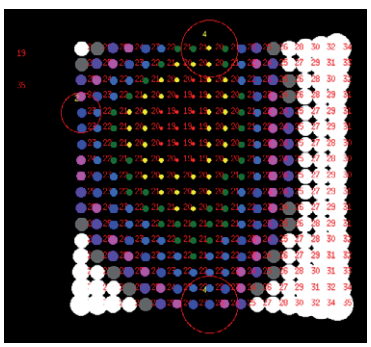


Fig-III-B1-7

Une autre approche consiste à se baser sur ce que l'on appelle la descente du gradient d'optimalité (DGO). Ce concept fait écho au nombre de Bell $\varpi(n)$, qui considère les valeurs successives de $S(n,k)$ en faisant varier 'k' de 1 à 'n'.

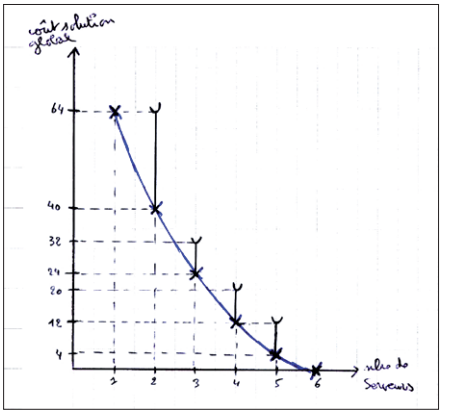


Fig-III-B1-8

Notre première étude empirique (partie III-A) nous a montré (cf. la figure *Fig-III-B1-8*) que, $\forall n$, nous avons $CG^*_{n,k} > CG^*_{n,k+1}$ avec $k \in [1,n]$. Autrement dit, plus le nombre de serveurs 'k' croît vers 'n' et plus le coût de la solution optimale $CG^*_{n,k}$ décroît vers 0.

Avant d'aller plus loin dans l'étude des *k-partitions* et des *k-positions*, nous allons faire quelques rappels terminologiques vus dans la partie III-A.

Rappels

$SOL_{n,k}$ est l'ensemble des solutions (sol_p) pour $SYS_{n,k}$ avec $sol_p=\{(conf_1, MM_{k-p})\}$ et $p \in [1, Card(SOL_{n,k})]$; $CG(sol_p)$ est le coût global de sol_p *(2)*.

$Conf_{q(n,k)}$ est l'ensemble des configurations $conf_l$ pour $SYS_{n,k}$ avec $l \in [1, Card(Conf_{q(n,k)})]$; chaque configuration $conf_l$ est une *k-partition* sur l'ensemble 'N' composé de 'n' éléments, appelés indifféremment «clients», points-motif (PM_i), ou points-source.

point-motif
point-source

$conf_l=\{Part_1, Part_2, \ldots, Part_k\}$, avec
$\{Part_1\} \cup \{Part_2\} \ldots \cup \{Part_k\}=N$, et
$\{Part_1\} \cap \{Part_2\} \ldots \cap \{Part_k\}=\{\varnothing\}$.

$Part_i=\{PM_{i-1}, PM_{i-2}, \ldots, PM_{i-ß_i}\}$ avec $i \in [1,k]$ et $ß_i$ le nombre de points-motif PM_i de $Part_i$.

POS_k est l'ensemble des multi-points MM_{k-p} avec $p \in [1, Card(POS_k)]$, chaque multi-points MM_{k-p} est une *k-position* sur la grille.

$MM_{k-p}=\{P_{p-1}, P_{p-2}, \ldots, P_{p-k}\}=\{(x_{p-1}, y_{p-1}), (x_{p-2}, y_{p-2}), \ldots, (x_{p-k}, y_{p-k})\}$.

Propriété: $conf_l$ est une *k-partition* «utile» si et seulement si il existe au moins une *k-position* de $SYS_{n,k}$, appelée multi-points MM_{k-p}, telle que $sol_p=\{(conf_1, MM_{k-p})\}$ et $CG(sol_p) < CG^*_{k-1}$. On dira qu'une *k-partition* «utile» est conforme à la descente du gradient d'optimalité (DGO).

On voit sur la figure *Fig-III-B1-9* des *k-partitions* qui ne sont pas des *k-partitions* «utiles». En effet trois *2-partitions* de la colonne centrale (k=2, n=6) ont CG=64. Cette valeur est la meilleure valeur pour ces trois *2-partitions* et elle est égale à $CG^*_{6,1}$ (en jaune dans la première colonne). L'inégalité n'est pas respectée donc ces *2-partitions* ne sont pas «utiles», contrairement à la première configuration de cette colonne centrale qui est l'optimale ($CG^*_{6,2}=38$).

Il en est de même pour les trois dernières *3-partitions* de la colonne de droite dont la valeur de CG respecte $CG \geq CG^*_{6,2}=38$.

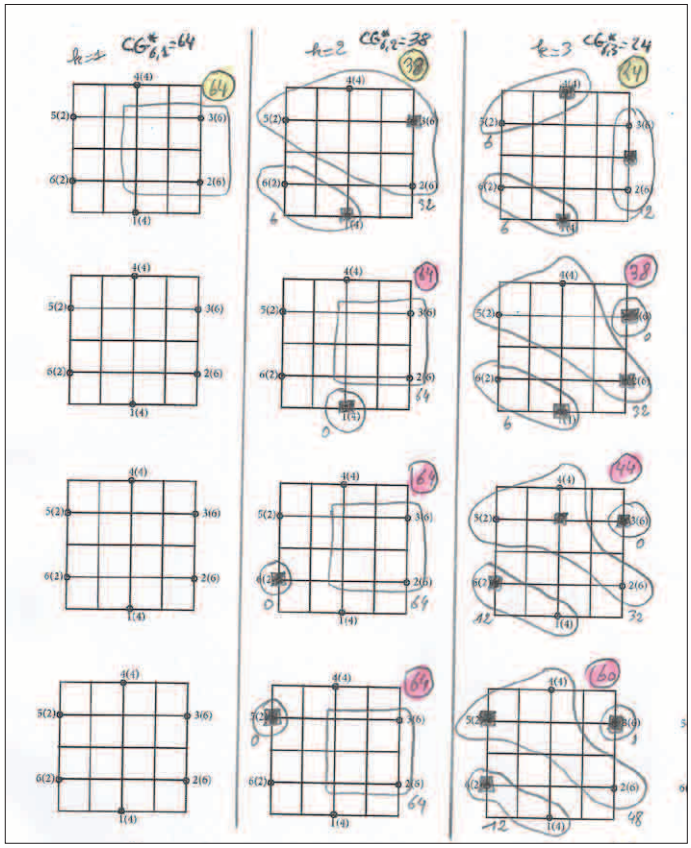


Fig-III-B1-9

Nous verrons, grâce aux calculs faits sur les solutions exhaustives (partie III-C), qu'il existe énormément de solutions «inutiles» détectées en appliquant la DGO.

$conf^*_l$

Définition: Une *k-partition* $conf_l$ est dite optimale, nous la noterons $conf^*_l$, si et seulement si il existe au moins une solution sol_p de $SYS_{n,k}$, telle que $sol_p=\{(conf_l,MM_{k-p})\}$ et $CG(sol_p)=CG^*_k$. Nous en déduirons que cette solution est elle-même optimale. Nous la noterons sol^*_p.
Sur la figure *Fig-III-B1-9* les trois *k-partitions* de la première ligne (jaune) sont toutes optimales respectivement pour k=1, k=2 et k=3.

sol^*_p

Propriété: Pour toute *k-partition* $conf_l$ de $Conf_{q(n,k)}$, il existe $Card(Conf_{q(n,k)})$ solutions $sol_p=\{(conf_l,MM_{k-p})\}$ ayant chacune une valeur pour $CG(sol_p)$ qui correspond au coût global de la solution. On peut trier les solutions suivant le résultat de CG.

CG_{min}

Propriété: Chaque *k-partition* $conf_l$ possède une valeur de CG minimale, notée CG_{min}, qui peut correspondre à une ou plusieurs *k-positions* sur la grille.

Nous proposons le mode opératoire suivant:

- pour les calculs exhaustifs des solutions sol_p et de la recherche de sol^*_p, nous trouverons les *k-partitions* utiles ($\psi(n,k)$) en fonction de la topologie utilisée, puis ne garderons que celles qui vérifient la DGO;
- pour les heuristiques de résolution que nous aborderons par la suite, nous proposerons un comportement qui explore les *k-positions* en ne parcourant que les *k-partitions* «utiles» afin de trouver la solution sol^*_p.

L'idée des heuristiques de résolution, que nous préciserons par la suite, est de déplacer les serveurs sur la grille pour explorer l'ensemble des *k-partitions* ($conf_l$) en appliquant aux 'n' clients (ou points-motif) la règle du «serveur le plus proche».

serveur le plus proche

Définition: La règle «du serveur le plus proche» appliqué à tous les 'n' clients de $SYS_{n,k}$ est définie par:
$\forall C_i \in CL$ on a $VU(C_i)=S_j \iff d(C_i,S_j)<d(C_i,S_h)$,
$\forall S_h \in SE$ avec $S_h \neq S_j$.

Cette règle préserve la notion de *k-partition* «utile» définie dans la partie III-A. En effet chaque client de $SYS_{n,k}$ choisira toujours le serveur le plus proche, qu'il soit sur la grille, ou en dehors dans le cas continu.

Cette règle garantit, pour toute solution $sol_p=\{(conf_1,MM_{k-p})\}$ de $SYS_{n,k}$, une intersection topologique nulle entre les partitions $Part_i$ et $Part_j$, deux à deux, de $conf_1$:

$\Omega(Part_i,Part_j)=\{\emptyset\}$, $\forall i,j\in[1,k]$ et $i\neq j$,
avec $conf_1=\{Part_1,Part_2,\ldots Part_k\}$.

Pour rappel, l'intersection topologique entre deux partitions $Part_i$ et $Part_j$ de $conf_1$, est le résultat de l'intersection (dans l'espace euclidien) des enveloppes convexes respectives des éléments (points-motif) de chaque partition $Part_i$ et $Part_j$ de $conf_1$, $\forall i,j\in[1,k]$.
En déplaçant les 'k' serveurs sur les *k-positions* (multi-points MM_{k-p}) de $SYS_{n,k}$, on n'explore que les *k-partitions* «utiles», réduisant considérablement la recherche de la solution optimale, conformément aux calculs théoriques proposés dans la partie III-A.
Cette approche heuristique nous amène à considérer la résolution comme une exploration dans l'espace des *k-partitions* en se basant sur un déplacement dont le vecteur est le multi-points et la cible la *k-partition* optimale ($conf^*_1$).

III-B2) De la grille au capteur : application à l'alphabet latin

La suite des calculs exhaustifs (exacts) va confimer l'intérêt de l'usage de DGO (descente du gradient d'optimalité) pour réduire la complexité en temps dans la recherche de la solution optimale de $SYS_{n,k}$.
Ceci va avoir une importance très grande dans le choix des heuristiques de résolution. Cela nous permet également d'étendre le rôle de la grille à celui de capteur. Chaque point de la grille peut être condidéré comme un capteur potentiel d'une source d'information reçue par $SYS_{n,k}$, qui lui est externe.

Propriété: Dans cette approche, chaque point-motif (PM_i) positionné sur la grille devient un point interface entre l'extérieur et l'intérieur de $SYS_{n,k}$. La grille devient alors un capteur capable d'une plasticité lui permettant de s'adapter à des sources externes diverses (au niveau de leur topologie).

capteur

Pour illustrer cette idée nous allons nous intéresser à l'alphabet latin. Nous verrons dans les parties suivantes comment cet exemple sera intéressant pour proposer une conceptualisation des mécanismes d'émergence de connaissances.
En parlant de topologie et en prenant le cercle comme exemple, nous avons pu définir une première approche pour trouver l'ensemble $\psi(n,k)$ des *k-partitions* «utiles». Cela présupposait que les clients étaient placés sur cette topologie. En généralisant la grille à la notion de capteur, les points-motif peuvent être placés sur la

grille conformément à la projection de l'objet étudié sur cette dernière.

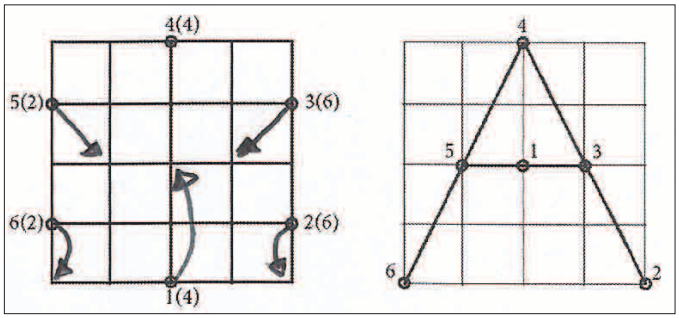


Fig-III-B2-10

Prenons le cas des lettres majuscules dans l'alphabet latin. Passer d'une topologie, par exemple le cercle, à une lettre se fait simplement en mettant les 'n' points-motif sur une projection de la forme étudiée (le 'A' sur la figure *Fig-III-B2-10*).

On peut utiliser plusieurs approches pour «sourcer» un objet étudié comme dans le cas de la reconnaissance de formes. On entend ici par sourcer un objet, le discrétiser par un ensemble de points (points-motif) sur la grille.

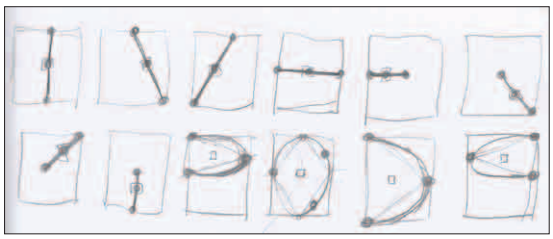


Fig-III-B2-11

Sur la figure *Fig-III-B2-10* il aura suffit de déplacer les six points-motif déjà présents sur la grille pour obtenir un sourçage de la lettre 'A'.

Si l'on se base sur les travaux de Georges Mounin[1] qui a proposé une décomposition en formes graphiques élémentaires des lettres de l'écriture latine pour les majuscules écrites en script, on voit que l'alphabet latin met en œuvre un système de douze traits graphiques minimaux, dont huit sont droits et quatre arrondis (cf. la figure *Fig-III-B2-11*). Il fait le constat que sur vingt-trois lettres, dix sont formées de trois traits graphiques, huit de deux traits, trois d'un seul trait et deux de quatre traits. On peut utiliser cette approche pour sourcer les lettres de l'alphabet.

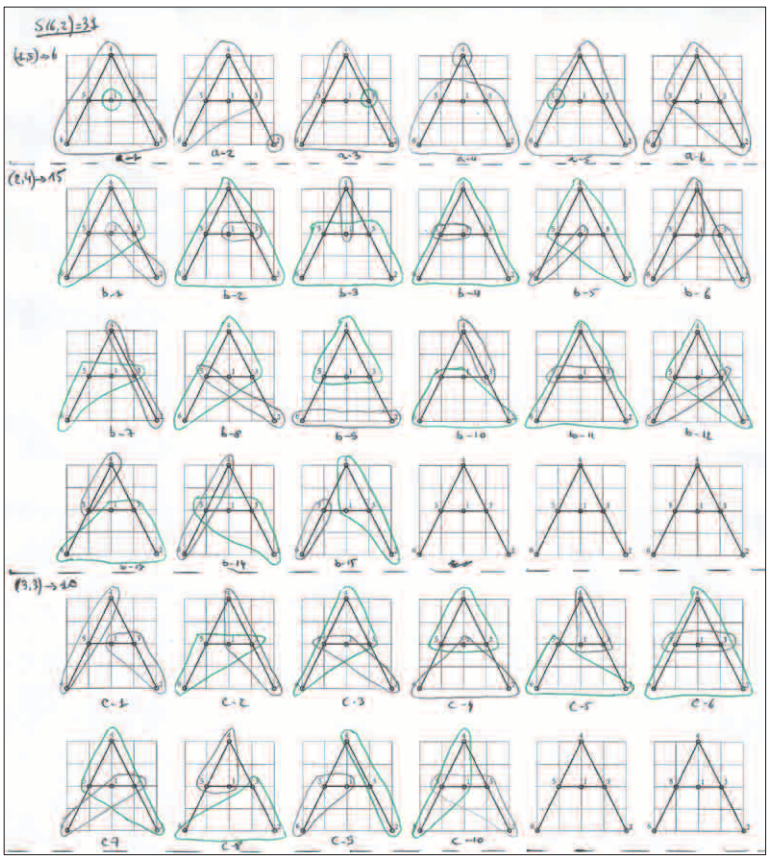


Fig-III-B2-12

Nous verrons par la suite comment on peut augmenter, sur la grille, le nombre de points ('n') pour sourcer de façon plus fine un même motif (par exemple 'A'). Nous étudierons l'impact de cette augmentation dans le processus de résolution (ici la représentation interne du motif).

Si nous appliquons à la lettre 'A' ce que nous avons vu dans la partie III-A nous pouvons identifier les différentes solutions sol_p sur notre capteur à six points-motif (cf/ la figure *Fig-III-B2-10*).

1 «*L'écriture script*», «*Introduction à la sémiologie*», éditions de Minuit, 1970, p.137-144.

Nous obtenons des résultats analogues (cf. la figure *Fig-III-B2-12*) à savoir :

- trente-et-une (S(6,2)) *2-partitions :* six pour $Conf_{1,5}$, quinze pour $Conf_{2,4}$ et dix pour $Conf_{3,3}$;
- douze ($\underline{\theta}(6,2)$) *2-partitions* «utiles» suivant l'intersection des enveloppes convexes des points-motif dans chaque partition : a-2, a-4, a-6, b-6, b-9, b-10, b-13, b-15, c-1, c-5, c-8 et c-9.

On peut également faire varier 'k', le nombre de partitions (serveurs). On peut en faire de même pour toutes les lettres de l'alphabet.

À faire

Nous allons construire des outils (partie III-C) nous permettant d'étudier toutes les lettres majuscules en script de l'alphabet latin, en paramétrant les dimensions de la grille ('D' et maille), le nombre 'n' des points-motif, que nous appelerons aussi points-source, et le nombre 'k' des serveurs. Le programme «calculCGstar.c», écrit en langage 'C', calcule, de façon exhaustive, pour un motif donné (par exemple la lettre 'A') - fourni sous la forme d'une liste de points-motif en x/y sur la grille - l'ensemble des multi-points possibles pour cette grille, en choisissant le nombre de partition ('k'). Ce programme calcule aussi le CG de chaque MM_{k-p} trouvé et place les MM_{k-p} et leur CG dans un fichier de résultats. Enfin ce programme identifie CG^* et sa *k-partition** (solution optimale pour $SYS_{n,k}$) associée.

La police de caractère National First Font Dotted (NFFD) propose l'alphabet avec des points-motif. Nous pourrons nous en servir pour faire des calculs exacts sur l'ensemble de l'alphabet et chercher à mettre en évidence des propriétés remarquables :

Travailler sur un système de représentation des connaissances, tout particulièrement sur celui de l'alphabet, pose la question du processus de différenciation permettant d'identifier, sans ambiguïté, une lettre par rapport à une autre. Nous reviendrons largement sur cette question par la suite, et nous verrons comment l'étude des *k-partitions*, en faisant varier 'k', sur une représentation donnée, peut participer à ce processus de différenciation indispensable à l'engrammage et donc à la reconnaissance et à la manipulation de connaissances. Concernant l'alphabet, nous pourrons jouer sur 'k' pour chercher les *k-partitions* d'un motif (par exemple le 'A') qui permettent de le signer, c.-à-d. de le différencier des autres motifs adjacents (les autres lettres de l'alphabet).

III-B3) L'ESPACE DES MULTI-POINTS MM_{k-p}

Nous avons défini précédemment (partie III-A) la notion de multi-points MM_{k-p}, appelé encore *k-position*, sur une grille donnée. Chaque association d'une *k-position* avec une *k-partition* $conf_l$ est une réponse sol_p au problème posé dans $SYS_{n,k}$. L'optimalité est représentée par la solution sol^*_p qui minimise CG pour $SYS_{n,k}$. On l'appelle $CG^*_{n,k}$.

Chaque *k-partition* $conf_l$ est composée de 'k' partitions disjointes: $conf_l=\{Part_1,Part_2,\ldots,Part_k\}$.

$\alpha=Card(POS_k)$

Propriété: Toute *k-partition* $conf_l$ de $SYS_{n,k}$, peut être associée à 'α' multi-points MM_{k-p} ($\alpha=Card(POS_k)$, *(7)*), conduisant aux solutions sol_p, avec $p\epsilon[1,\alpha]$, dont certaines sont «utiles», d'autres, en plus d'être «utiles», sont «optimales» pour cette *k-partition* (CG_{min}):

$sol_1=\{(conf_1,MM_{k-1})\}$, $CG(sol_1)=cg_1$,
$sol_2=\{(conf_1,MM_{k-2})\}$, $CG(sol_2)=cg_2$,
…,
$sol_\alpha=\{(conf_1,MM_{k-\alpha})\}$, $CG(sol_\alpha)=cg_\alpha$.

Ces 'α' multi-points MM_{k-p} peuvent être ordonnés de façon croissante en fonction de $cg_i=CG(sol_i)$, avec $i\epsilon[1,\alpha]$.

Nous pouvons appliquer aux multi-points associés à une *k-partition* $conf_l$, la notion de multi-points «optimal» et de multi-points «utile», comme nous l'avons fait précédemment pour les *k-partitions*.
On dira que le MM_{k-h} de $conf_l$ est optimal si sa solution (sol_p) associée, a un CG minimal.
De même tout MM_{k-h} de $conf_l$ sera un multi-points «utile» si sa solution (sol_p) associée aura un cg inférieur au CG de la meilleure solution avec k-1. Cette notion d'utilité sera particulièrement intéressante dans la recherche d'une caractérisation des motifs étudiés (par exemple les lettres).

multi-points «optimal» MM^*_{k-p}

Définition: Un multi-points MM_{k-h} associé à une *k-partition* $conf_l$ de $SYS_{n,k}$, composée elle-même de 'α' multi-points MM_{k-p}, est considéré comme un multi-points «optimal» pour $conf_l$, et noté MM^*_{k-h}, si la solution sol_h associée vérifie:

$sol_h=\{(conf_1,MM_{k-h})\}$, avec $h\epsilon[1,\alpha]$ et
$CG(sol_h)<CG(sol_i)$ $\forall i\epsilon[1,\alpha]$, avec
$i\neq h$ et $sol_i=\{(conf_1,MM_{k-i})\}$.

multi-points «utile»

Définition: Un multi-points MM_{k-h} associé à une *k-partition* $conf_l$ de $SYS_{n,k}$ est considéré comme un multi-points «utile» pour $conf_l$ si la solution sol_h associée vérifie:

$CG(sol_h)<CG^*_{n,k-1}$.

Propriétés: Toute *k-partition* «utile» $conf_l$ possède au moins un multi-points «utile».

Nous pouvons préciser la définition de $conf^*_1$, déjà donnée, en partant de MM^*_{k-p}.

Définition: Une *k-partition* $conf_1$ de $SYS_{n,k}$ est dite «optimale», et notée $conf^*_1$, si son multi-points «optimal» MM^*_{k-p}, associé à sol_p, vérifie: $CG(sol_p)=CG^*_{n,k}$.

MULTI-POINTS «UTILES» ET «OPTIMAUX»: UN EXEMPLE

Prenons la lettre 'A' définie avec vingt-cinq points-motif (n=25) sur une grille de taille D=6, avec deux partitions (k=2, *2-partition*), comme le montre la figure *Fig-III-B3-13*.
Nous ferons une étude complète de cette lettre 'A' après avoir vu, dans la partie suivante, les outils nécessaires pour faire des calculs exhaustifs. En attendant cette étude nous pouvons prendre quelques résultats de ces calculs, pour illustrer les notions de multi-points «utile» et de multi-points «optimal».
Comme le montre la figure *Fig-III-B3-13* nous avons deux points (*k-position* avec k=1) MP^*_{1-1} et MP^*_{1-2} qui sont optimaux pour une seule partition avec:

$CG^*_{25,1}=46.29$, $MP^*_{1-1}=\{(2,3)\}$ et $MP^*_{1-2}=\{(3,3)\}$.

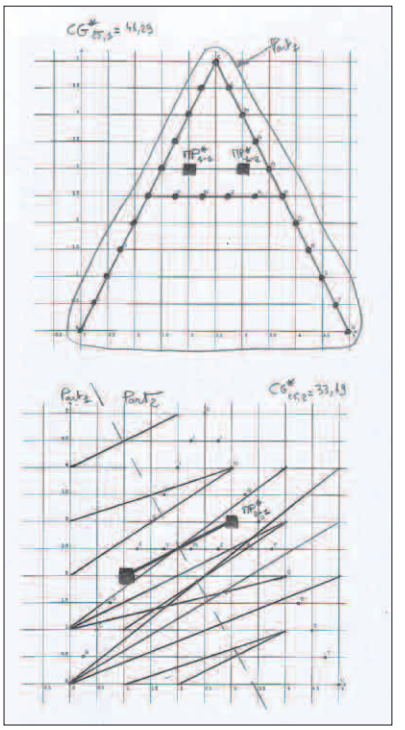


Fig-III-B3-13

La symétrie de la lettre 'A', et des vingt-cinq points-motif qui la définissent, explique la présence de ces deux points optimaux. Pour les mêmes raisons de symétrie de 'A' il en sera de même avec les *2-positions* où nous aurons deux bi-points (multi-points avec k=2) optimaux et donc deux *2-partitions* optimales. Nous n'en étudierons qu'une.
Nous avons:

$sol^*_p=\{(conf_1, MP^*_{2-1})\}$,
$conf_1=\{Part_1, Part_2\}$,
$VOYANT(S_1)=\{A,B,C,D,E,F,G,V\}=Part_1$,
$VOYANT(S_2)=\{H,I,J,K,L,M,N,O,P,Q,R,S,T,U,W,Z,A1\}=Part_2$.

Sur la figure *Fig-III-B3-13* nous avons visualisé les treize premiers bi-points en représentant le bi-points «optimal» MM^*_{2-1} en traits noir épais, les autres étant avec des traits plus clairs. La ligne pointillée délimite les deux partitions $Part_1$ et $Part_2$.
Nous avons les résultats suivant:

$CG^*_{25,2}=\underline{33.69}$ et $MP^*_{2-1}=\{(1,2),(3,3)\}$,
$CG_{25,2}=\underline{38.73}$ et $MP_{2-2}=\{(1,1),(4,3)\}$,
$CG_{25,2}=\underline{40.29}$ et $MP_{2-3}=\{(0,1),(4,3)\}$,
$CG_{25,2}=\underline{41.11}$ et $MP_{2-4}=\{(0,1),(4,2)\}$,
$CG_{25,2}=\underline{41.28}$ et $MP_{2-5}=\{(0,2),(3,4)\}$,
$CG_{25,2}=\underline{45.10}$ et $MP_{2-6}=\{(0,3),(3,4)\}$,
$CG_{25,2}=\underline{46.14}$ et $MP_{2-7}=\{(0,1),(4,4)\}$,
$CG_{25,2}=53.24$ et $MP_{2-8}=\{(1,0),(4,1)\}$,
$CG_{25,2}=54.70$ et $MP_{2-9}=\{(0,0),(5,2)\}$,
$CG_{25,2}=54.77$ et $MP_{2-10}=\{(0,0),(5,3)\}$,
$CG_{25,2}=56.51$ et $MP_{2-11}=\{(2,0),(4,1)\}$,
$CG_{25,2}=59.99$ et $MP_{2-12}=\{(0,0),(5,4)\}$,
$CG_{25,2}=66.36$ et $MP_{2-13}=\{(0,4),(2,5)\}$.

Le gradient d'optimalité est bien respecté puisque : $CG^*_{25,1}(=46,29) > CG^*_{25,2}(=33,69)$.

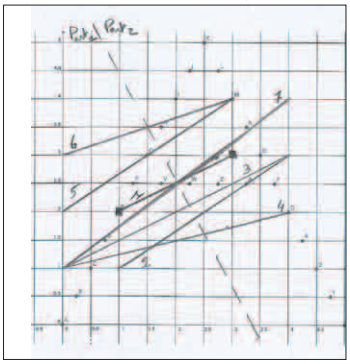


Fig-III-B3-14

Nous avons donné les treize premiers bi-points $MM_{2\text{-}p}$ par ordre croissant de leur CG associé ; il en existe bien d'autres, six-cent-trente (36!/(2!(36-2)!)), en tout. On voit sur cet exemple (cf. la figure *Fig-III-B3-14*) que seuls les sept premiers bi-points (de $MP^*_{2\text{-}1}$ à $MP_{2\text{-}7}$) sont des bi-points «utiles». En effet la valeur du CG associé à chacun de ces bi-points est inférieure à celle de $CG^*_{25,1}$.
Il ne sera pas intéressant, au moins dans un premier temps, de travailler sur les bi-points qui ne sont pas «utiles».

PRINCIPE DU DÉPLACEMENT DES SERVEURS SUR LA GRILLE

De la même façon qu'il existe un gradient d'optimalité entre les valeurs de $CG^*_{n,k}$ avec 'k' (nombre de serveurs) variant de 1 à 'n' (le nombre de points-motif), il en existe un autre dans chaque partition $Part_i$ d'une *k-partition* $conf_1$ donnée, conduisant au puits de $Part_i$, c.-à-d. l'endroit sur la grille où la valeur de CG est minimale pour $Part_i$.

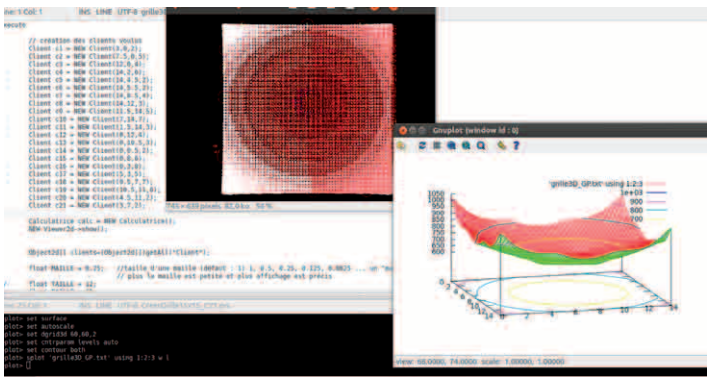


Fig-III-B3-15

Sur la figure *Fig-III-B3-15* nous avons visualisé le gradient d'optimalité d'une partition $Part_i$ possédant vingt-et-un clients (n=21). Le gradient apparaît sur le schéma de droite ; il a été généré en 3D par *gnuplot*, avec le puits correspondant à la zone sur la grille où CG est le plus faible (optimum local).
Peut-on parler de l'espace des multi-points $MM_{k\text{-}p}$ et en définir les propriétés (topologie, gradient, puits, distance,…) ?

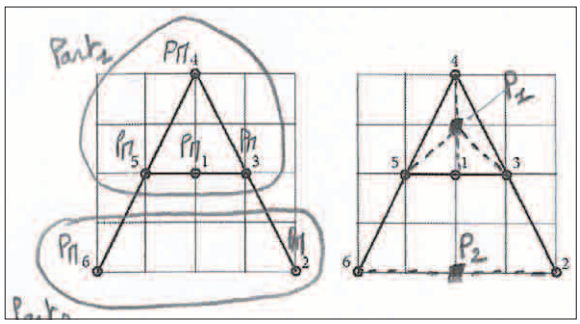


Fig-III-B3-16

En se basant sur le principe du «serveur le plus proche», évoqué plus haut à propos des heuristiques résolutives, cette résolution consiste à déplacer continuement les serveurs de *k-position* en *k-position* à travers l'espace des *k-partitions*.

Si l'on reprend la lettre 'A', sourcée par six points-motif (cf. la figure *Fig-III-B2-10*), et qu'on lui applique la solution sol_p suivante (cf. la figure *Fig-III-B3-16*) avec deux serveurs S_1 et S_2, respectivement sur les points P_1 et P_2, nous obtenons :

$sol_p=\{(conf_1, MM_{2\text{-}p})\}$,
$conf_1=\{Part_1, Part_2\}$,
$MM_{2\text{-}p}=\{P_1, P_2\}$,
$VOYANT(S_1)=\{PM_1, PM_3, PM_4, PM_5\}=Part_1$,
$VOYANT(S_2)=\{PM_2, PM_6\}=Part_2$.

À chaque déplacement du serveur S_1 et/ou du serveur S_2 sur la grille nous changeons de *k-position* et donc de solution (de sol_p à sol_p') pour $SYS_{n,k}$, mais pas forcément ni de *k-partition*, ni la valeur CG associée. On peut avoir $CG(sol_p)=CG(sol_p')$.

Espace et déplacements des multi-points MM_{k-p}

MP^*_{2-1} associé à $conf_1$, représentant la lettre 'A' composée de vingt-cinq points-motif placés sur une grille/capteur de trente-six points, définit la solution optimale (sol^*_p) à $SYS_{25,2}$. Dans l'espace des bi-points associé à $SYS_{25,2}$, MP^*_{2-1} représente un puits (CG minimale), comme cela peut être représenté sur la figure *Fig-III-B3-15*.
Dans la mesure où l'on cherche ce puits (attracteur) en déplaçant les serveurs sur des *2-positions* (bi-points), regardons comment on peut se déplacer dans l'espace des bi-points afin d'arriver dans le puits. De façon opérationnelle il suffit de descendre le gradient d'optimalité à l'intérieur de l'espace des bi-points d'une *2-partition* $conf_1$ donnée.
Se déplacer dans l'espace des multi-points dépend de la grille. Si l'on prend par exemple une grille carrée dans laquelle chaque point possède un *4-voisinage* (quatre points), tout multi-points de type *k-position* possède 'k' points ayant chacun d'eux, quatre points voisins.

voisinage

Définition: Un multi-points MM_{k-i} est voisin d'un multi-points MM_{k-j} sur une grille en *q-voisinage* si chacun des points $P_{j-1}, P_{j-2}, \ldots, P_{j-k}$ de MM_{k-j} est dans le *q-voisinage* de façon exclusive de l'un des points $P_{i-1}, P_{i-2}, \ldots, P_{i-k}$ de MM_{k-i}.

L'exclusivité ici nous impose des voisinages deux à deux, tous différents.

Propriété: Tout multi-points MM_{k-i} possède $(q+1)^k$ multi-points voisins sur une grille en *q-voisinage*.

On peut également définir une distance entre deux multi-points MM_{k-i} et MM_{k-j} dans l'espace des multi-points de $SYS_{n,k}$.

distance

$\gamma(MM_{k-i}, MM_{k-j})$

Définition: La distance entre deux multi-points MM_{k-i} et MM_{k-j} dans l'espace des multi-points de $SYS_{n,k}$, nommée $\gamma(MM_{k-i}, MM_{k-j})$, est le nombre de multi-points inclus dans la chaîne (sans cycle) de voisinage la plus courte pour passer de MM_{k-i} à MM_{k-j} (en se déplaçant d'unité en unité sur la grille), en excluant MM_{k-i} et en incluant MM_{k-j}. Deux multi-points voisins sont à une distance de 1 ($\gamma(MM_{k-i}, MM_{k-j})=1$) l'un de l'autre dans l'espace des multi-points.

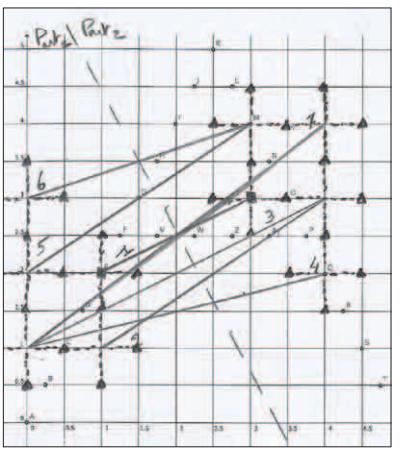


Fig-III-B3-17

Dans le cas des bi-points sur un *4-voisinage* (cf. la figure *Fig-III-B3-17*), on en déduit qu'un bi-points possède $5^2=25$ bi-points voisins.
Nous pouvons adopter cette notion de *4-voisinage* à l'exemple des figures *Fig-III-B2-17/18*.

Le tableau ci-après montre, pour les multi-points suivants, les distances respectives deux à deux :

γ	MP_{2-1}	MP_{2-2}	MP_{2-3}	MP_{2-4}	MP_{2-5}	MP_{2-6}	MP_{2-7}
MP_{2-1}	0	**2**	4	4	**2**	4	4
MP_{2-2}	**2**	0	**2**	**2**	4	6	**2**
MP_{2-3}	4	**2**	0	**2**	4	4	**2**
MP_{2-4}	4	**2**	**2**	0	6	6	4
MP_{2-5}	**2**	4	4	6	0	**2**	**2**
MP_{2-6}	4	6	4	6	**2**	0	4
MP_{2-7}	4	**2**	**2**	4	**2**	4	0

graphe

Définition : On peut représenter l'espace des multi-points d'une *k-partition* $conf_1$, par un graphe valué (chaque arc valant 1) non orienté, puisque $\gamma(MM_{k-i},MM_{k-j})=\gamma(MM_{k-j},MM_{k-i})$, appelé graphe des distances entre multi-points de $conf_1$.

Le graphe des distances entre multi-points d'une *k-partition* $conf_1$ nous permet de décrire l'espace dans lequel se déplacent les multi-points «utiles» de $conf_1$ et la façon dont ces déplacements peuvent être mis en œuvre.

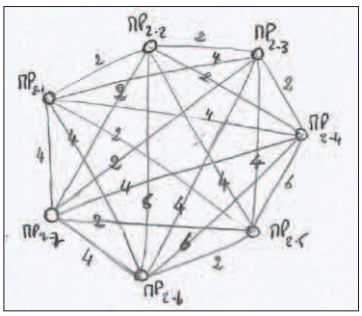


Fig-III-B3-18

Dans la mesure où les différents serveurs, représentant les points du multi-points, peuvent se déplacer simultanément, comme ce sera le cas dans nos simulations, nous devrons introduire des mécanismes de synchronisation entre ces déplacements dans les heuristiques de résolution de $SYS_{n,k}$.

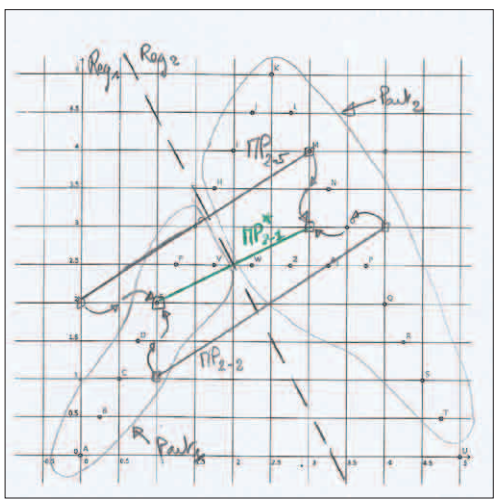


Fig-III-B3-19

région

DÉPLACEMENTS DES MM_{k-p} ET *K-PARTITION*

Maintenant nous allons étudier les conséquences des déplacements des multi-points vis-à-vis des *k-partitions* de $SYS_{n,k}$. Comme nous l'avons déjà évoqué plus haut, passer de MM_{k-i} à MM_{k-j} peut ou non avoir des conséquences sur la recherche de la solution.

Sur la figure *Fig-III-B3-19*, le passage de MP_{2-2} ou MP_{2-5} à MP_{2-1} ne change pas la *k-partition* $conf_1$, mais change la solution pour $SYS_{25,2}$. Nous passons de sol_p à sol^*_p, avec $CG(sol_p)>CG(sol^*_p)$ puisque sol^*_p est la solution optimale pour $SYS_{25,2}$.

Toujours sur la figure *Fig-III-B3-19*, nous avons fait apparaître en pointillé la séparation de la grille en deux régions (Reg_1 et Reg_2) qui délimitent les zones respectives des deux points de chaque bi-points impliqué dans la résolution de $SYS_{25,2}$.

Pour rappel, nous avons :

$sol^*_p=\{(conf_1,MP^*_{2-1})\}$,
$MP^*_{2-1}=\{P_{1-1},P_{1-2}\}=\{(1,2),(3,3)\}$,
$conf_1=\{Part_1,Part_2\}$, avec
$VOYANT(S_1)=\{A,B,C,D,E,F,G,V\}=Part_1$ et
$VOYANT(S_2)=\{H,I,J,K,L,M,N,O,P,Q,R,S,T,U,W,Z,A1\}=Part_2$.

Propriété: Toute *k-partition* $conf_1$ découpe la grille en 'k' surfaces disjointes deux à deux, appelées régions (Reg_k). Chaque région accueille un sous-ensemble des points-motif de $SYS_{n,k}$ formant une partition ($Part_i$) ; elle est parcourue par un et un seul serveur (point du multi-points associé) à la fois.

Reg_k

Sur une grille, une région correspond à un ensemble connexe de points de cette grille, alors qu'avec un espace continu (euclidien à deux dimensions, par exemple) une région s'apparente à une surface connexe.

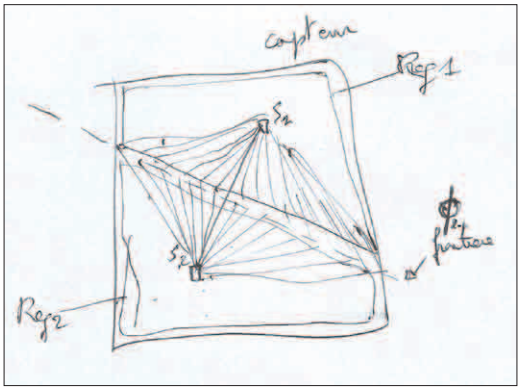


Fig-III-B3-20

Définition: On appelle i^e région de la grille, que l'on nomme Reg_{k-i}, avec $i \epsilon [1,k]$, la surface de la grille dans laquelle se trouve le i^e point P_{q-i} du bi-points courant MP_{k-q}, lorsque ce dernier se déplace.

Propriété: Tant que tous les points du multi-points courant restent dans leur région respective, alors la *k-partition* courante reste inchangée. Chaque déplacement de l'un des points du multi-points, entraîne une nouvelle solution sol_p avec sa valeur CG associée.

Dans ces conditions deux régions adjacentes ou voisines possèdent une frontière commune.

Définition: On appelle frontière du capteur courant pour son bi-points MM_{2-p}, composé des points P_{p-1} et P_{p-2}, la perpendiculaire au segment $\underline{P_{p-1}P_{p-2}}$ qui passe par le centre de ce segment. On la nomme Φ_{2-p}.

frontière, Φ_{2-p}

Propriété: La frontière Φ_{2-p} du bi-points MM_{2-p}, est l'ensemble des points du capteur à égale distance des deux points P_{p-1} et P_{p-2} de MM_{2-p}. Tout point du capteur en dehors de Φ_{2-p} sera plus proche de l'un des points de MM_{2-p}. Il sera dans la région Reg_{2-i} correspondante.

Un bi-points MM_{2-p} produit (est associé à) une *2-partition* ($conf_1$) à partir de sa frontière Φ_{2-p}.

Propriété: Tous les bi-points MM_{2-p} qui ont la même frontière Φ_{2-p} (dont chaque point appartient à l'une des régions correspondantes), produisent la même *2-partition* ($conf_1$).

Cette notion de production d'une *2-partition* donnée est importante, puisqu'elle correspond à la relation qu'il existe entre une *2-partition* ($conf_1$) donnée et l'ensemble de ses MM_{2-p} associés.

Dans la suite de cette étude on ne considèrera dans cet ensemble que les 'β' MM_{2-p} «utiles» associés à $conf_1$, avec $\beta<\alpha$, et $\alpha=Card(POS_k)$ *(7)*.
Chacun de ces 'β' MM_{2-p} «utiles» conduira à calculer une solution sol_p spécifique pour $conf_1$. On pourra ordonner ces solutions et donc les MM_{2-p} associés, suivant la valeur $CG(sol_p)$.

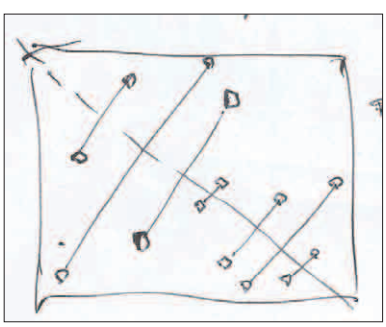


Fig-III-B3-21

β

En effet sur la figure *Fig-III-B3-13* nous avions visualisé les treize meilleurs bi-points, alors que sur la figure *Fig-III-B3-14* nous n'avions gardé que les sept bi-points «utiles».
Or si l'on regarde les six bi-points «non-utiles» (MP_{2-8} à MP_{2-13}) on s'apperçoit que certains d'entre eux (MP_{2-8}, MP_{2-11} et MP_{2-13}) sont parallèles à l'un des bi-points «utiles», alors que certains bi-points «utiles» peuvent également être parallèles entre eux (MP_{2-2} et MP_{2-5}) :

$MP^{*}_{2-1}=\{(1,2),(3,3)\}$ // $MP_{2-11}=\{(2,0),(4,1)\}$ // $MP_{2-13}=\{(0,4),(2,5)\}$,
$MP_{2-2}=\{(1,1),(4,3)\}$//$MP_{2-5}=\{(0,2),(3,4)\}$,
$MP_{2-3}=\{(0,1),(4,3)\}$,
$MP_{2-4}=\{(0,1),(4,2)\}$,
$MP_{2-6}=\{(0,3),(3,4)\}$//$MP_{2-8}=\{(1,0),(4,1)\}$,
$MP_{2-7}=\{(0,1),(4,4)\}$,
$MP_{2-9}=\{(0,0),(5,2)\}$,
$MP_{2-10}=\{(0,0),(5,3)\}$,
$MP_{2-12}=\{(0,0),(5,4)\}$.

Avant d'aborder le problème de l'espace des *k-partitions* (dans la partie III-B4), afin de pouvoir passer d'une configuration quelconque $conf_1$ à la configuration optimale $conf^{*}_1$, nous allons regarder comment, en déplaçant un MM_{2-p}, nous pouvons rester dans la $conf_1$ associée ou en sortir ($conf_1$').
Nous venons de voir que pour rester dans une configuration $conf_1$ donnée, il suffit de passer d'un bi-points à un autre de telle sorte qu'ils respectent la frontière Φ_{2-p} commune. Dans la mesure où ce sont l'ensemble des 'β' bi-points MM_{2-p} «utiles» qui caractérisent une *k-partition* $conf_1$ donnée, il faut maintenant considérer non plus une seule frontière Φ_{2-p}, mais l'ensemble des frontières associées aux 'β' MM_{2-p} «utiles». Sur l'exemple de la figure *Fig-III-B3-14* seuls six bi-points «utiles» sur sept ont des frontières différentes, deux d'entre eux étant parallèles. Comme nous le montre la figure *Fig-III-B3-22*, tout bi-points («utile» ou non) qui respecte l'une de ces six frontières produira la même configuration $conf_1$.

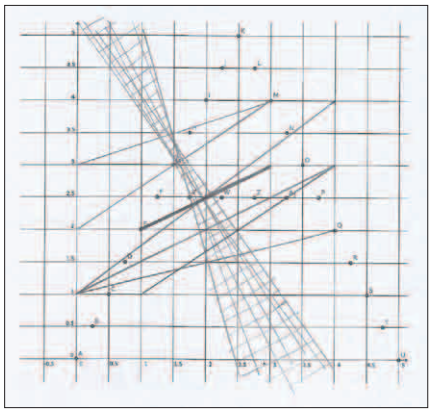


Fig-III-B3-22

On peut ainsi générer, via ces six frontières Φ_{2-p}, des bi-points MM_{2-p} qui produisent $conf_1$. La zone hachurée, sur la figure *Fig-III-B3-22*, matérialise le faisceau de production de $conf_1$.

faisceau de frontières

Définition: On appelle faisceau de frontières pour une configuration $conf_1$ donnée, l'ensemble des frontières Φ_{2-p} générées par les multi-points MM_{2-p} «utiles» de $conf_1$.

Lorsque que nous déplaçons un bi-points MM_{2-p}, nous pouvons le faire soit en déplaçant simultanément ses deux points, soit un à la fois. Au-delà de ces possibilités apparaît fondamentalement deux modes de déplacement des serveurs d'un multi-points. Soit les serveurs se déplacent en autonomie les uns par rapport aux autres (mode «marche aléatoire»), soit leurs déplacements sont dépendants les uns des autres (mode «synchronisé»). Le premier mode aléatoire conduit le multi-points de configuration en configuration, alors que le second (synchronisé) permet de rester dans une configuration donnée. On parle d'un déplacement qui vérifie un invariant de configuration.

mode marche aléatoire
mode synchronisé

Plaçons-nous, comme le montre la figure *Fig-III-B3-23*, dans le cas (mode aléatoire) d'un bi-points où seul l'un des deux points est déplacé ($P_{p-1}\rightarrow P_{p-1}'$). Nous avons :

$MM_{2-p}=\{P_{p-1},P_{p-2}\}$, avec $P_{p-1}\subset Reg_1$ et $P_{p-2}\subset Reg_2$,
puis
$MM_{2-p}'=\{P_{p-1}',P_{p-2}\}$, avec $P_{p-1}'\subset Reg_1'$ et $P_{p-2}\subset Reg_2'$.

Nous sommes passé des régions Reg_1 et Reg_2 aux régions Reg_1' et Reg_2' et de la frontière Φ_{2-p} à la nouvelle frontière Φ_{2-p}', avec :
$Reg_a=Reg_1\cap Reg_1'$ et $Reg_b=Reg_2\cap Reg_2'$.

On peut voir, sur la figure *Fig-III-B3-23*, que la configuration courante $conf_1$ est préservée ou non lorsque l'on déplace $P_{p-1}\rightarrow P_{p-1}'$ suivant que les points-motif appartiennent ou non aux régions Reg_a et Reg_b.

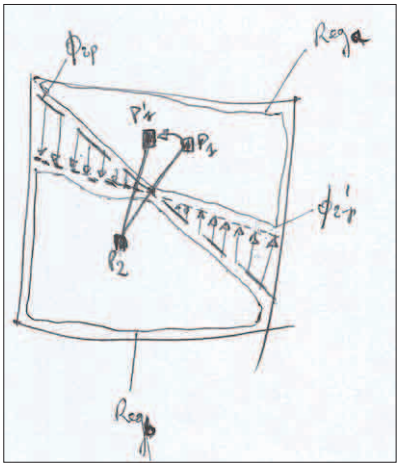


Fig-III-B3-23

Propriété: Si tous les points-motifs PM_i de $conf_1$ sont soit dans Reg_a soit dans Reg_b - régions résultant de l'intersection de l'ancienne région avec la nouvelle suite au déplacement de l'un des points du bi-points - alors cette configuration courante $conf_1$ est préservée. Sinon nous passons de $conf_1$ à $conf_1'$ avec $conf_1\neq conf_1'$.

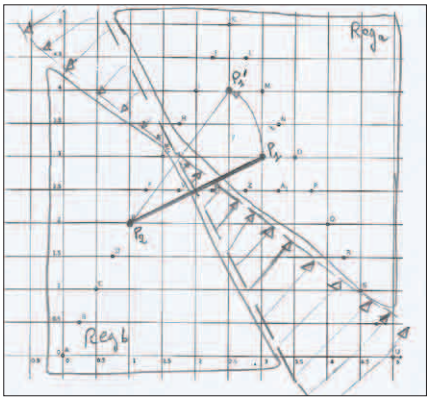


Fig-III-B3-24

Cette propriété (à démontrer) est vérifiée avec les MM_{2-p} «utiles» de notre exemple, comme le montre la figure *Fig-III-B3-22* avec la zone hachurée. En effet cette zone hachurée délimite les régions Reg_a et Reg_b dans lesquelles se trouvent tous les points-motifs de la configuration courante $conf_1$. On peut ainsi prendre deux à deux chaque frontière des MM_{2-p} «utiles» et l'on aura toujours la propriété ci-dessus vraie.
En revanche il existe des situations qui ne vérifient pas cette propriété, comme celle sur la figure *Fig-III-B3-24*. En déplaçant P_{p-1} vers P_{p-1}' nous voyons que le point-motif 'W' appartient à la zone hachurée, ce qui correspond au fait que ce point-motif est passé d'une région à une autre et n'appartient ni à Reg_a, ni à Reg_b, entraînant ainsi un changement de configuration ($conf_1\rightarrow conf_1'$).

On peut généraliser cette propriété avec un nombre quelconque 'k' de régions Reg_i. Nous pouvons illustrer cette généralisation par la figure *Fig-III-B3-25* dans laquelle nous avons trois régions (k=3). En déplaçant $P_{p-1}\rightarrow P_{p-1}'$ nous obtenons :
$P_{p-0}\rightarrow P_{p-0}'$, centres des cercles passant respectivement par $(P_{p-1},P_{p-2},P_{p-3})$ et $(P_{p-1}',P_{p-2},P_{p-3})$,
$Reg_1\rightarrow Reg_1'$, $Reg_2\rightarrow Reg_2'$, $Reg_3\rightarrow Reg_3'$
avec Reg_a, Reg_b et Reg_c telles que
$Reg_a=Reg_1\cap Reg_1'$, $Reg_b=Reg_2\cap Reg_2'$ et $Reg_c=Reg_3\cap Reg_3'$.

Les points-motif 'J','M','N' et 'W' étant dans la zone hachurée, nous sommes passés de la configuration courante $conf_1$ à une autre configuration $conf_1'$. Si aucun point-motif n'était

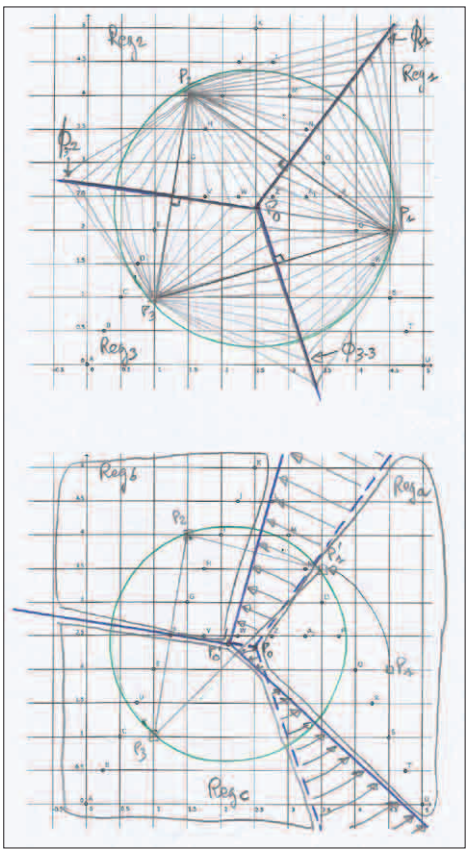


Fig-III-B3-25

dans cette zone hachurée nous aurions eu conservation de la configuration courante $conf_1$.

Sur la figure *Fig-III-B3-26* nous avons fait bouger simultanément les trois points P_{p-1}, P_{p-2} et P_{p-3} pour obtenir les points P_{p-1}', P_{p-2}' et P_{p-3}'.
Le centre P_{p-0}' du nouveau cercle qui inscrit les trois points résultants est à l'intersection des trois régions Reg_a, Reg_b et Reg_c. Là encore on constate que les points-motifs 'F','V','W','Z','M' et 'N' sont dans la zone hachurée, faisant ainsi passer de $conf_1$ à $conf_1'$.

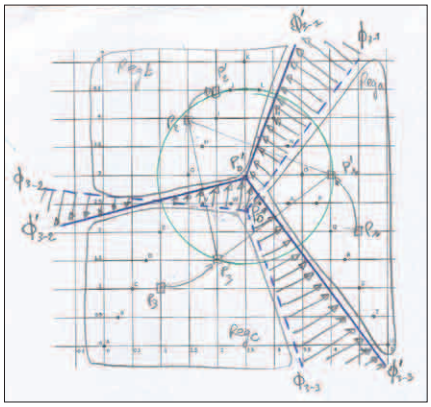


Fig-III-B3-26

Nous avons évoqué la notion de puits pour chaque partition $Part_i$ de $conf_1$, représentant la position du serveur S_j dans $Part_i$ sur la grille pour laquelle $cg(sol_p,S_j)$ est minimale. On peut transposer cette notion de puits aux régions Reg_a, Reg_b et Reg_c, de telle sorte que tant que les serveurs S_j se déplacent sans remettre en question $conf_1$, ils peuvent suivre le gradient d'optimalité qui va les conduire à la solution optimale pour la configuration courante $conf_1$. D'où l'intérêt de la deuxième approche qui consiste à déplacer les serveur tout en préservant l'invariant de configuration.

Nous venons de voir qu'il y a deux modes (aléatoire et synchronisé) de déplacement des serveurs S_j sur la grille, qui peuvent être combinés, afin de trouver la solution optimale (sol^*_p) à $SYS_{n,k}$. Quand les serveurs se déplacent sans changer de configuration $conf_1$ (en respectant le faisceau de frontières associé), ils peuvent suivre le gradient d'optimalité de $conf_1$, qui les conduit vers son optimum local ou l'optimum global, si $conf_1$ est la configuration optimale $conf^*$.
Le mode aléatoire ne va pas permettre de rester dans la configuration courante (ou à de rares exceptions en jouant sur la configuration des points-motif comme le montre la figure *Fig-III-B3-23*), alors que le mode synchronisé le permettra.

L'idée du mode synchronisé est de déplacer les serveurs de multi-points en multi-points ayant tous une frontière commune dans le faisceau (cf. la figure *Fig-III-B3-22*) de frontières associé (frontières Φ_{2-p} des multi-points MM_{2-p} «utiles» de $conf_1$).

De façon opératoire le mode synchronisé peut être mis en œuvre assez facilement. Sans anticiper sur la partie heuristique de résolution de ce travail, nous pouvons déjà esquisser les principes qui régissent ce mode de déplacement des multi-points sur la grille. Ces principes ne sont pas sans rappeler les règles de Reynolds dans le cas de la modélisation des vols d'oiseaux ou autres «boids»[1].

1 C. Reynolds, «Flocks, herds and schools: A distributed behavioral model», in Proceedings of the 14th annual conference on Computer graphics and interactive techniques, SIGGRAPH'87, p.25-34, 1987.

Nous avons trois principes :

1. éloignement dans la limite de la grille ;
2. direction commune ;
3. un comportement commun : la symétrie inversée ou le «vol du mirroir» avec évitement/ collision.

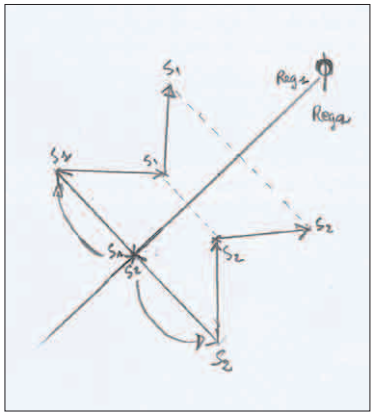


Fig-III-B3-27

Comme l'illustre la figure *Fig-III-B3-27*, les deux serveurs S_1 et S_2 peuvent ainsi démarrer au même endroit sur la grille (cela permet de prendre en compte des algorithmes de type division cellulaire…). Ils vont ensuite combiner 1) et 3). En s'éloignant l'un de l'autre dans le respect de 1) (la limite de la grille) et de 3) (de façon symétrique : même vitesse) ils vont fixer une direction commune 2), qui correspond à la frontière (axe perpendiculaire au segment de départ et passant en son milieu, c.-à-d. le point de départ des deux serveurs) $\Phi_{2\text{-}p}$ définie précédemment. En continuant avec 3) et le respect de 1) (dimensions de la grille) ils vont naturellement préserver la direction commune 2). Ce faisant les deux serveurs vont suivrent les multi-points qui préservent la configuration $conf_1$ courante (règle d'invariance).

Avec cette méthode entièrement décentralisée (sans contrôle central) les serveurs vont pouvoir explorer tous les multi-points qui préservent $conf_1$, qu'ils se trouvent sur la grille (mode discret) ou dans le plan euclidien (mode continu). En y ajoutant le principe de gradient d'optimalité local à $conf_1$, ils pourront trouver le puits et donc le minimum local.

Nous allons maintenant aborder la question du changement de configuration $conf_1$ (*k-partition*), ou comment passer d'une configuration $conf_1$ à une autre configuration $conf_1$'. En effet, jusqu'à présent, nous avons regardé comment trouver l'optimum local à une configuration donnée, sachant qu'il en existe de nombreuses dont une seule (ou quelques unes dans des cas particulier de symétrie du problème $SYS_{n,k}$) qui est optimale.

III-B4) L'ESPACE DES *K-PARTITIONS* $CONF_L$

Nous allons regarder les propriétés de l'espace des configurations (*k-partitions*) afin de comprendre quels mécanismes mettre en œuvre pour retrouver la configuration optimale ($conf^*_1$) à partir de n'importe quelle configuration $conf_1$ de $SYS_{n,k}$.

Définition : Dans le cas ou un déplacement de l'un des serveurs (point d'un $MM_{k\text{-}p}$) sur la grille fait changer de *k-partition* ($conf_1$ vers $conf_h$ sur la figure *Fig-III-B4-28*), alors ces 2 *k-partitions* ($conf_1$ et $conf_h$) sont dites voisines dans l'espace des *k-partitions*.

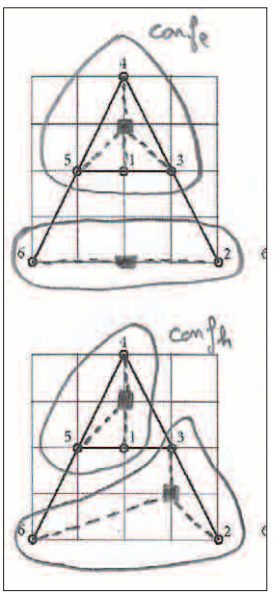


Fig-III-B4-28

On peut également définir une métrique dans cet espace.

$\delta(conf_l, conf_h)$

Définition: La distance entre deux *k-partitions* $conf_l$ et $conf_h$ dans l'espace des *k-partitions* de $SYS_{n,k}$, nommée $\delta(conf_l, conf_h)$, est le nombre de *k-partitions* incluses dans la chaîne (sans cycle) de transformation pour passer de $conf_l$ à $conf_h$, moins 1. Deux *k-partitions* voisines sont à une distance 1 ($\delta(conf_l, conf_h)=1$) l'une de l'autre dans l'espace des *k-partitions*.

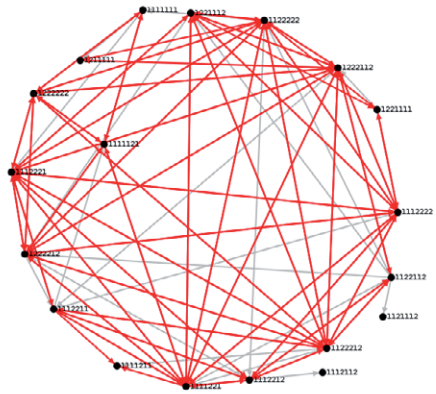


Fig-III-B4-29

On pourra ainsi construire le graphe d'adjacence des *k-partitions* dans l'espace des *k-partitions* de $SYS_{n,k}$.
Sur la figure *Fig-III-B4-29* nous avons le graphe d'adjacence des dix-neuf *2-partitions* «utiles» sur les vingt-et-une théoriques ($\underline{\theta}(7,2)=21$) de C_7S_2 (n=7 et k=2), obtenues avec le simulateur *oRis* et le logiciel *GraphStream*.

Sur les figures *Fig-III-B3-25* nous avons trois points et nous regardons ce qui se passe quand l'un d'entre eux se déplace. Dans l'exemple mentionné, ce déplacement occasionne un changement de configuration. Si l'on associe la notion de configuration $conf_l$ à celle d'un état du système $SYS_{n,k}$, alors les trois points ont changé d'état ($conf_l \rightarrow conf_l'$) alors que deux d'entre eux n'ont pas bougé sur la grille.

À l'inverse les trois points auraient pu bouger sans pour autant changer d'état. En revanche quand un point reste fixe tout en changeant d'état, il change de frontière. Si l'on regarde le système par le prisme des déplacements des points des multi-points on peut parler d'une composante quantique du système. En effet un point peut être simultanément (avec une certaine probabilité) dans différents états du système en fonction de ce que deviennent les autres points du multi-points courant.

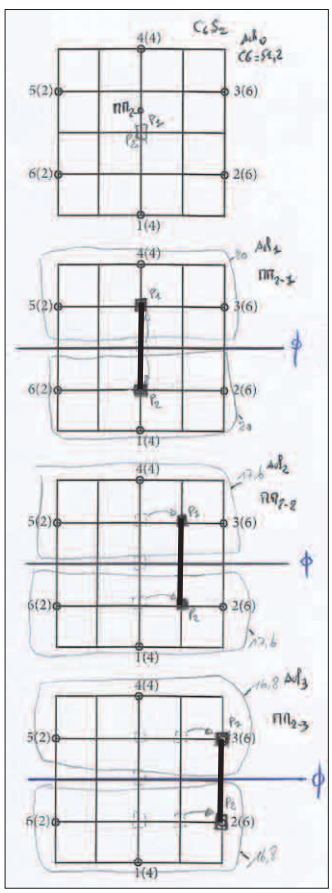


Fig-III-B4-30

Nous allons illustrer les changements d'états (configuration $conf_l$) en prenant deux exemples. le premier sera issu de C_nS_k avec n=6 et k=2, alors que le second reprendra l'exemple de la lettre 'A' avec vingt-cinq points-motif (n=25) et deux régions (k=2).

Exemple de déplacements des MM_{2-p} en mode aléatoire

Sur la figure *Fig-III-B4-30* nous avons C_6S_2 avec comme positions initiales ($conf_0$) des deux serveurs le bi-points MM_{2-0} constitué des points P_1 et P_2 au centre de la grille. La solution sol_0 obtenue est logiquement loin d'être optimale: $CG(sol_0)=51,2$.
En appliquant le premier principe d'éloignement aux deux serveurs ($MM_{2-0} \rightarrow MM_{2-1}$), nous faisons apparaître une première frontière Φ_{2-1} correspondant à la configuration $conf_1$, avec $CG(sol_1)=20+20=40$.

Nous déplaçons à nouveau les serveurs (MM_{2-1}→MM_{2-2}). Cela ne change pas la frontière Φ_{2-1}, le nouveau bi-points MM_{2-2} vérifie les critères de segment perpendiculaire à la frontière avec le passage de cette dernière par le milieu du segment. Nous conservons donc $conf_1$, mais progressons pour la solution sol_2 : $CG(sol_2)=35,2$.
Enfin nous déplaçons encore les serveurs (MM_{2-2}→MM_{2-3}) toujours dans l'idée du respect de la frontière existante. Nous gardons $conf_1$ en améliorant la solution sol_3 obtenue : $CG(sol_3)=33,6$.
Nous allons poursuivre les déplacements des serveurs en dérogeant à la règle de l'invariance de la configuration courante, pour sortir de $conf_1$ afin d'aller vers la configuration optimale $conf^*_1$ ($conf_3$ sur notre exemple).

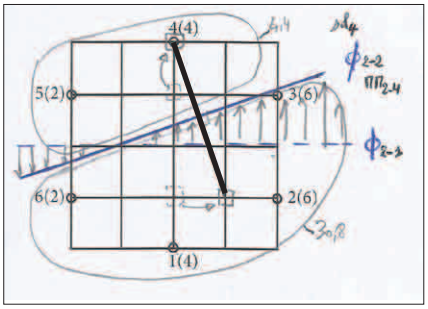


Fig-III-B4-31

Cette fois (cf. la figure *Fig-III-B4-31*) S_1 monte d'une unité alors que S_2 va d'une unité vers la droite (MM_{2-3}→MM_{2-4}). Nous obtenons une nouvelle frontière Φ_{2-2} et par conséquent une nouvelle configuration $conf_2$ avec sol_4 moins bonne que sol_3 : $CG(sol_4)=35,2$.

S_1 et S_2 se déplacent d'une unité vers la droite (1re planche de la figure *Fig-III-B4-32*). Nous repassons dans $conf_1$ avec une nouvelle solution sol_5 et $CG(sol_5)=35,6$.

S_1 reste fixe et S_2 monte d'une unité (2e planche de la figure *Fig-III-B4-32*). Nous repassons dans $conf_2$ avec une nouvelle solution sol_6 et $CG(sol_6)=35,6$.

Enfin S_1 part à gauche d'une unité et S_2 descend d'une unité (3e planche de la figure *Fig-III-B4-32*). Nous arrivons à la solution optimale sol_7 avec $conf_3$ et $CG(sol_7)=32,6$.

Nous avons traversé les étapes suivantes pour arriver à la solution optimale (sol_7) : $conf_0(sol_0)$→$conf_1(sol_1)$→$conf_1(sol_2)$→$conf_1(sol_3)$→$conf_2(sol_4)$→$conf_1(sol_5)$→$conf_2(sol_6)$→$conf_3(sol_7)$.

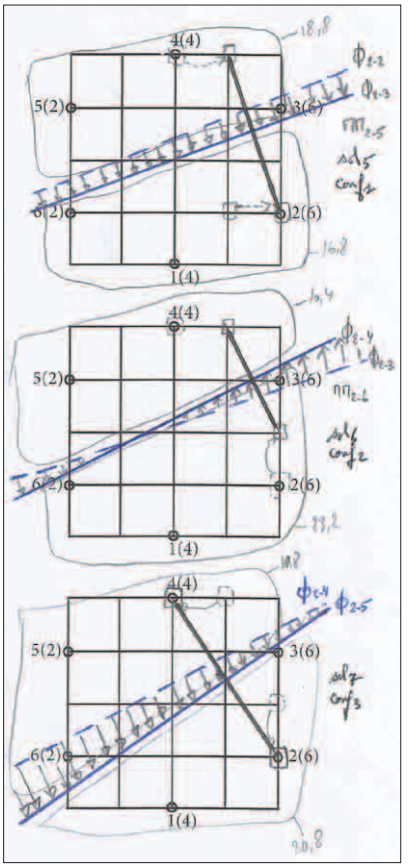


Fig-III-B4-32

Pour compléter le mode aléatoire d'exploration de l'espace des configurations ($conf_1$) de $SYS_{n,k}$, qui s'apparente à celui d'une marche aléatoire, nous pouvons reprendre l'exemple précédent à partir de la solution sol_4. Dans cette approche les serveurs se déplacent indépendamment les uns des autres au risque de changer de configuration sans converger vers la solution optimale ou de tomber dessus rapidement.
C'est le cas sur la figure *Fig-III-B4-33* où S_2 va d'une unité vers la droite et tombe directement sur la solution optimale ($conf_3$, sol_7 avec $CG(sol_7)=32,6$).

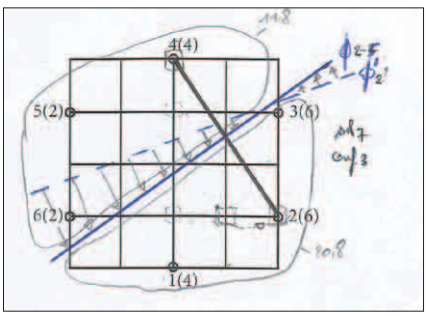


Fig-III-B4-33

Dans ce cas nous avons traversé les étapes suivantes : $conf_0(sol_0) \rightarrow conf_1(sol_1) \rightarrow conf_1(sol_2) \rightarrow conf_1(sol_3) \rightarrow conf_2(sol_4) \rightarrow conf_3(sol_7)$.

On peut proposer bien d'autres exemples d'explorations, même sur un exemple aussi simple que C_6S_2, comme celui des figures précédentes.
Et voici (cf. les figures *Fig-III-B4-34/35*) deux autres explorations possibles pour lesquelles le passage de $conf_1$ à $conf_3$ se fait par $conf_2$.

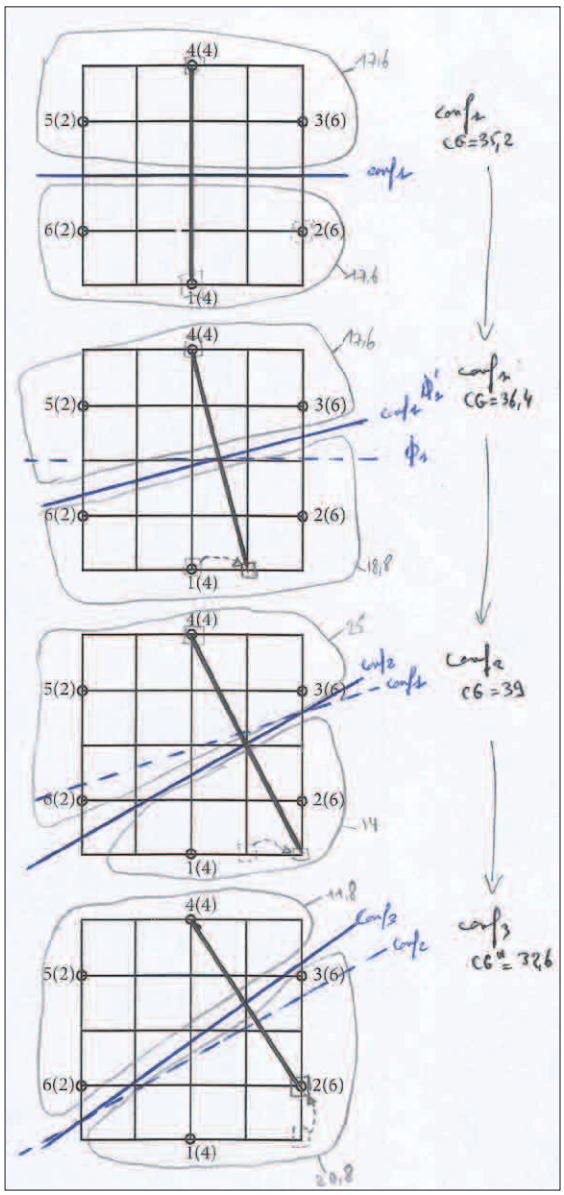


Fig-III-B4-34

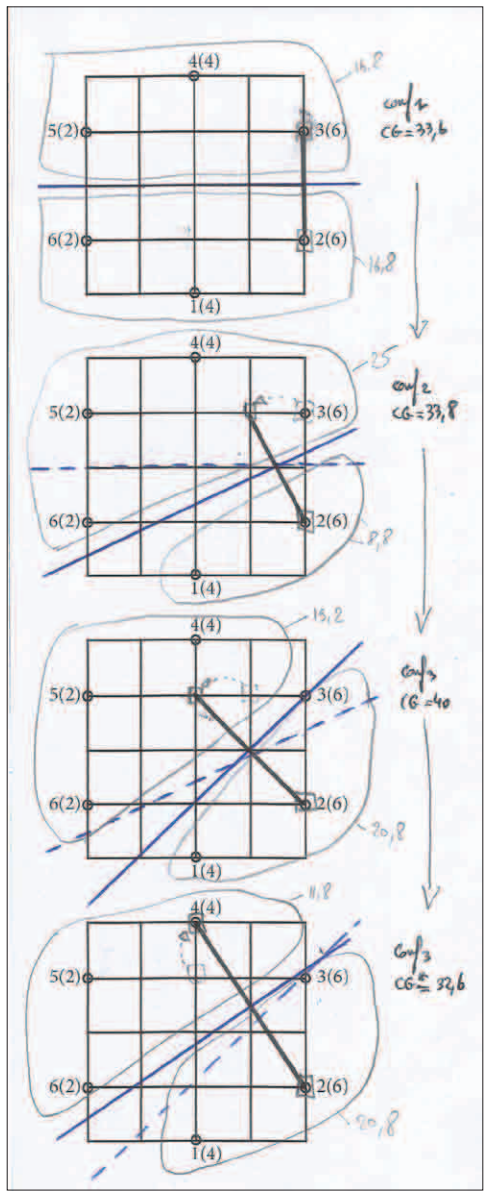


Fig-III-B4-35

EXEMPLE DE DÉPLACEMENTS DES MM_{2-p} EN MODE SYNCHRONE

Nous repartons de la solution sol_3 de l'exemple précédent (cf. la figure *Fig-III-B4-30*) et appliquons le principe de la synchronisation entre S_1 et S_2. Dans cette première solution les serveurs S_1 et S_2, respectivement sur les points P_{p-1} et P_{p-2} de la grille, ont pour frontière Φ_{2-1} et définissent la configuration $conf_1$, avec $CG(sol_3)=33,6$.
Nous allons regarder le cheminement du bi-points courant MM_{2-p} sur la grille, dans la recherche de la solution optimale (sol_7) en mode synchrone.

Sur la figure *Fig-III-B4-36*, la 1re planche montre en bleu la frontière $\Phi_{2\text{-}1}$ de sol_3, dans le cadre de la configuration $conf_1$, mais aussi en rouge pointillé la frontière $\Phi_{2\text{-}7}$ de la solution optimale sol_7 avec $conf_3$, solution que nous cherchons à atteindre.

Sur la 2e planche (cf. la figure *Fig-III-B4-36*) les serveurs S_1 et S_2, respectivement sur les points $P_{p\text{-}1}$ et $P_{p\text{-}2}$ de la grille, se décalent vers la gauche sur des parallèles à $\Phi_{2\text{-}1}$ et de façon symétrique (mêmes distances parcourues) jusqu'à atteindre, par le milieu du segment reliant $P_{p\text{-}1}$ à $P_{p\text{-}2}$, le point d'intersection entre la frontière courante $\Phi_{2\text{-}1}$ et la frontière recherchée $\Phi_{2\text{-}7}$.

Nous appelons ce point la «$porte_{1\text{-}3}$» entre les configurations $conf_1$ et $conf_3$.

Sur la 3e planche (cf. la figure *Fig-III-B4-36*) S_1 et S_2 opèrent une sorte de rotation (déplacement symétrique) autour de la $porte_{1\text{-}3}$, pour recaler leur segment sur la frontière $\Phi_{2\text{-}7}$ recherchée: sol_5' dans $conf_3$ avec $CG(sol_5')=38,2$. La distance parcourue par chaque serveur est identique (vers la gauche pour S_1 et vers la droite pour S_2). Cette phase est charnière dans le déplacement des serveurs, car elle permet de passer d'une configuration à une autre tout en se calant sur la nouvelle frontière. C'est par ce mécanisme que les serveurs vont pouvoir explorer l'espace des configurations et donc, indirectement, celui des solutions. Ils pourront utiliser les gradients d'optimalitéslocaux à chaque partition.

Sur la figure *Fig-III-B4-37* les serveurs «glissent» vers la solution tout en respectant la frontière $\Phi_{2\text{-}7}$. La solution obtenue (sol_6') est toujours dans $conf_3$ avec $CG(sol_6')=38,2$.

Enfin, sur la figure *Fig-III-B4-38* les serveurs arrivent sur sol_7, la solution optimale.
Précisons la notion de «porte» entre deux configurations.

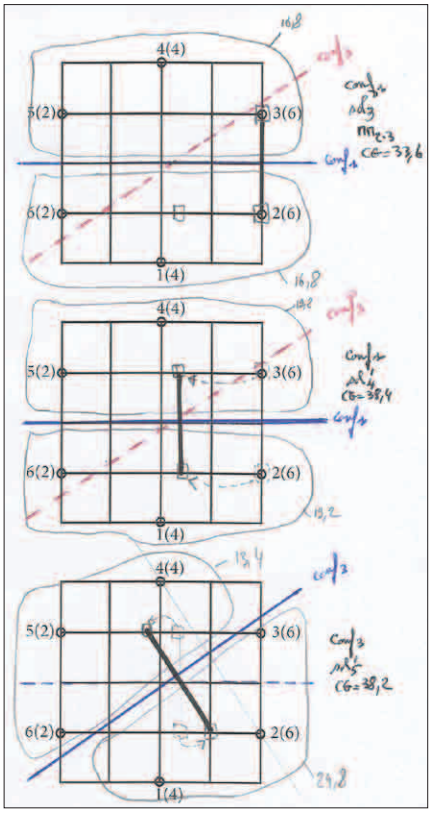


Fig-III-B4-36

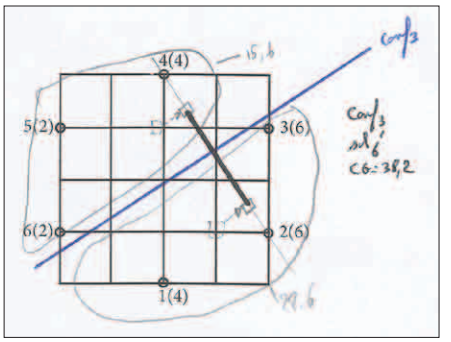


Fig-III-B4-37

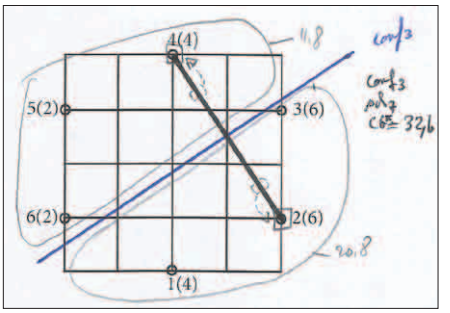


Fig-III-B4-38

LA «PORTE» OU LE POINT DE PASSAGE ENTRE CONFIGURATIONS

Dans les déplacements en mode synchronisé nous appliquons les trois principes (cf. la figure *Fig-III-B3-27*) apparentés à ceux des «boids» de C. Reynolds. Ces principes garantissent de façon décentralisée que des serveurs autonomes préservent l'invariance de configuation, lors de leurs déplacements. Or, nous venons de voir dans l'exemple précédent, que la recherche de l'optimal doit garantir cette invariance pour tirer partie des gradients d'optimalité locaux, tout en étant

capable de changer de configuration pour se rapprocher de la configuration optimale.
La 2e planche de la figure *Fig-III-B4-36* nous illustre le passage de $conf_1$ vers $conf_3$, sans passer par $conf_2$, comme ce fût le cas dans l'exemple en mode aléatoire (cf. la figure *Fig-III-B4-34/35*).
L'intersection entre deux frontières joue un rôle très important pour le passage d'une configuration à une autre et donc dans la notion de configurations adjacentes ou voisines.

$porte_{i-j}$

Définition: Nous appelons «$porte_{i-j}$» des configurations $conf_i$ et $conf_j$ du système $SYS_{n,k}$, le ou les points d'intersection entre les différentes frontières appartenant aux faisceaux de frontières respectifs de ces deux configurations.

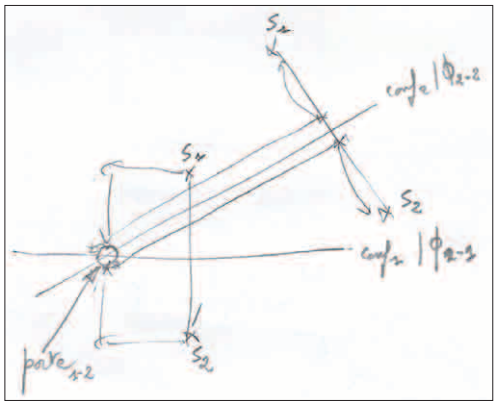


Fig-III-B4-39

La porte, ou point de passage, entre deux configurations, permet de respecter les gradients d'optimalité respectifs aux deux configurations. Le changement d'axe s'opère de façon symétrique par les serveurs du multi-points et accompagne le passage d'une configuration à une autre. Un jeu existe dès lors que l'on peut passer (micro-changements d'axe) sur n'importe quelle frontière du faisceau correspondant.
Il existe plusieurs cas de figure. En particulier il se peut que l'intersection entre les frontières de deux configurations ne soit pas placées sur le capteur. Dans ce cas la porte est inaccessible. On ne peut pas conclure pour autant sur l'adjacence des configurations concernées. En effet, dans pareil cas, un déplacement aléatoire peut faire passer quand même d'une configuration à l'autre, les rendant adjacentes l'une de l'autre.

Propriété: S'il existe sur le capteur/grille une intersection entre deux frontières respectivement issues des faisceaux de frontières des configurations $conf_i$ et $conf_j$, alors ces configurations sont adjacentes.

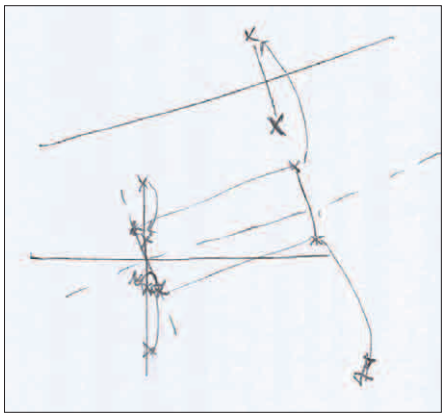


Fig-III-B4-40

Propriété: Si entre deux frontières $\Phi_{2\text{-}i}$ et $\Phi_{2\text{-}j}$ parallèles il n'y a pas de point-motif, alors ces deux frontières seront liées à la même configuration $conf_1$.

Si l'on opère un changement d'axe (cf. la figure *Fig-III-B4-40*) pour se caler d'une frontière $\Phi_{2\text{-}i}$ sur une frontière $\Phi_{2\text{-}j}$ en dehors du point de passage (c'est forcément le cas lorsque l'intersection des frontières est hors grille), cela ne va pas marcher. Le bi-points désaxé sera bien parallèle au bi-points cible, mais on ne pourra jamais atteindre ce dernier en mode synchronisé.

La figure *Fig-III-B4-41* montre les dix-sept premiers multi-points $MM_{2\text{-}p}$ (par ordre croissant de $CG(sol_p)$) et les frontières $\Phi_{2\text{-}p}$ associées (en pointillé bleu, sauf pour la frontière de $conf^*_1$ qui est en trait bleu plein). Les taches jaunes représentent les intersections entre frontières qui se situent sur le capteur, alors qu'en rouge on voit celles qui sont à l'extérieur du capteur. Chaque intersection représente une «$porte_{i\text{-}j}$» entre deux configurations adjacentes $conf_i$ et $conf_j$. Dans cet exemple, où le motif capté est la lettre 'A' sur une grille avec D=6 et vingt-cinq points-motif, nous avons douze configurations $conf_1$ différentes pour dix-sept $MM_{2\text{-}p}$. On voit que si l'on voulait tirer parti de toutes les portes $porte_{i\text{-}j}$ pour explorer l'espace des configurations de $SYS_{n,k}$, il faudrait soit agrandir le capteur, soit faire un zoom arrière du motif sur le capteur.

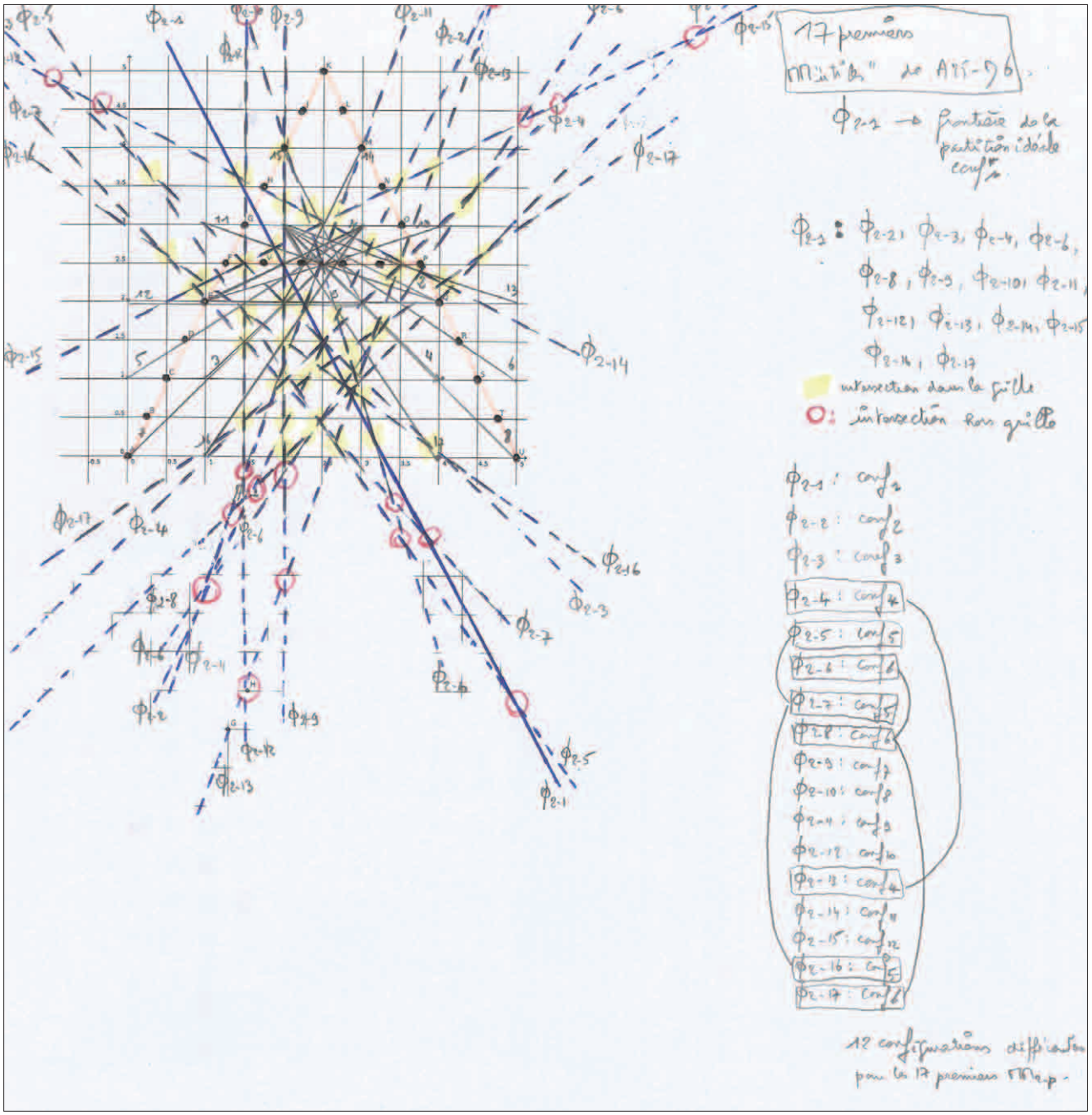

17 premiers
Φ2-1 → frontière de la partition idéale conf°
Φ2-2 : Φ2-2, Φ2-3, Φ2-4, Φ2-6, Φ2-8, Φ2-9, Φ2-10, Φ2-11, Φ2-12, Φ2-13, Φ2-14, Φ2-15, Φ2-16, Φ2-17
intersection dans la grille
intersection hors grille
Φ2-1 : conf 2
Φ2-2 : conf 2
Φ2-3 : conf 3
Φ2-4 : conf 4
Φ2-5 : conf 5
Φ2-6 : conf 6
Φ2-7 : conf 5
Φ2-8 : conf 6
Φ2-9 : conf 7
Φ2-10 : conf 8
Φ2-11 : conf 9
Φ2-12 : conf 10
Φ2-13 : conf 4
Φ2-14 : conf 11
Φ2-15 : conf 12
Φ2-16 : conf 5
Φ2-17 : conf 6
12 configurations différentes pour les 17 premiers Φ2-p

Fig-III-B4-41

Outils pour le calcul des solutions exactes

III-C

Cette partie est en hommage à tous les étudiants (en moyenne cent-cinquante par an, soit environ trois-mille-cinq-cents au total) que j'ai eu pendant mes vingt-cinq années d'enseignement du « Système » et plus particulièrement de la programmation « Système en 'C' et en shell sur *Linux* », dans le département informatique de l'I.U.T. de Caen.

Nous allons aborder ici les outils que nous avons développés pour compléter cette étude théorique, dont les approfondissements sont basés sur une exploration exhaustive de cas réels. Nous avons choisi l'alphabet latin en lettres majuscules. Ces outils sont tout aussi adaptés à toutes autres sortes de motifs.
Devant faire face à des calculs dont la complexité algorithmique en temps est très vite gigantesque, nous avons du trouver une méthodologie pour les réaliser de façon incrémentale. En plus de cette méthodologie, nous avons du parallèliser, autant que faire se peut, ces calculs pour aller plus loin dans leur profondeur.
Cette approche à plusieurs intérêts. Elle permet de calculer des solutions exactes au problème $SYS_{n,k}$, et plus particulièrement à ceux de type C_nS_k. Nous pouvons ainsi valider les résultats théoriques de notre étude. Au-delà de ça, et surtout, elle permet de piloter et de valider les heuristiques de résolution à $SYS_{n,k}$ que nous proposons dans la suite de ce travail.

III-C1) Calcul des *k-partitions* «utiles» et de CG^*

La première série de programmes *calculCGstar*, écrits en langage 'C', permet de calculer les *k-partitions* d'un système $SYS_{n,k}$ et les valeurs $CG(sol_p)$ des solutions sol_p correspondantes. On calcule également les valeurs optimales $CG^*_{n,k}$. Cinq versions du même programme ont été réalisées afin d'ajouter progressivement de nouvelles fonctionalités et en améliorer certaines déjà existantes[1].
Travaillant sur un capteur qui a la forme d'une grille carrée avec une largeur de 'D' unités et une *maille* définissant la finesse de la grille, nous avons intégré ces paramètres dans nos programmes.
Les points-motif du motif traité (par exemple vingt PM_i pour la lettre 'A') sont décrits dans le fichier «lettre.data» grâce à une ligne par PM_i avec ses coordonnées (x/y) sur la grille.
Sur la figure *Fig-III-C1-1* nous avons un exemple de motif et ses points-motif associés, représenté dans le logiciel *GeoGebra*.

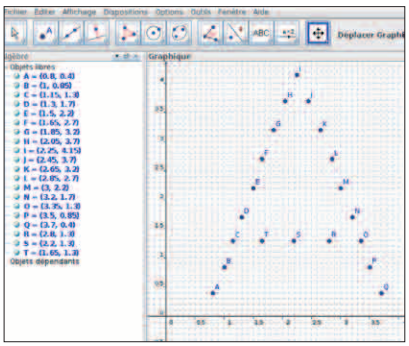


Fig-III-C1-1

Pour optimiser les codes, dans l'espace mémoire nécessaire à l'exécution du programme, nous avons créé une structure générique (*struct ptslet*) qui sera allouée dynamiquement (*malloc*) en fonction des besoins :

```
typedef struct ptslet {
  unsigned int nb_lignes;
  unsigned int nb_colonnes;
  unsigned short **coeff;
}PtsLet;
```

1 J'ai écrit un article qui s'intitulait «An experiment in prototyping using the object model as structuring agent». Il a paru dans les actes de la conférence «Tools'90 Technology of Object-oriented Languages and Systems», Paris, 1990, éditions Angkor Paris. Il décrivait une façon incrémentale de produire du code que m'avaient inspiré mes diverses expériences de codage. J'étais alors au début de ma carrière de recherche. Plus de trente années après, j'utilise toujours cette méthode pour mes programmes, mais au-delà du codage, elle me sert également pour rédiger ce testament de recherche.

Nous avons besoin de créer les points-motif, les points de la grille et les points des multi-points (*k-position*) MM_{k-p}. Dans les trois cas nous avons des structures *PtsLet* avec les particularités suivantes :

1. *points-motif (xyLet)* : autant de lignes que de points-motif, avec pour chaque ligne deux coordonnées (x/y) pour les coordonnées des points-motif et 'z' pour stocker l'indice du point le plus proche dans le vecteur du multi-points MM_{k-p} associé ;
2. *points de la grille (xyzGrille)* : D^2 lignes, avec pour chaque ligne deux coordonnées (x/y) pour les coordonnées des points de la grille et 'z' pour gérer les doublons (points déjà traités) et les MM_{2-p} ayant deux coordonnées identiques ;
3. *multi-points* MM_{k-p} *(Mpoints)* : 'k' lignes, avec pour chaque ligne deux coordonnées (x/y) pour les coordonnées des points de MM_{k-p} et 'z' pour stocker le CG local au point considéré.

calculCGstar-v1

Cette version (cent-soixante lignes, commentaires inclus) traite uniquement les bi-points ou *2-positions* (MM_{2-p}) qui séparent la solution en deux régions (k=2). Nous devons entrer la dimension 'D' de la grille. Nous utilisons l'allocation dynamique de mémoire (*malloc*) pour la matrice (*PtsLet xyLet*) des points-motif. On lance le calcul des MM_{2-p} qui sont sauvegardés dans le fichier «Mpoints.txt».
Nous allouons dynamiquement la matrice (*PtsLet xyzGrille*) représentant la grille (capteur).
Le calcul des MM_{2-p} est réalisé par deux boucles imbriquées qui parcourent chacune tous les points de la grille. Dans le cas de D=6 (dimension du côté de la grille), la maille est ici d'une unité, avec k=2 nous obtenons six-cent-trente multi-points MM_{2-p} (7). Pour chaque MM_{2-p} trouvé on calcule $CG(sol_p)$ en fonction du positionnement des points-motif sur la grille.
Un handler capture le signal *SIGINT* (CTRL-C) afin de connaître, à tout moment des calculs, leur état d'avancement (exprimé en pourcentage du calcul complet).
Une version (*calculCGstar-v1b*) du programme a été faite sur le même principe que *calculCGstar-v1*, mais uniquement pour les *3-positions* (MM_{3-p}).

calculCGstar-v2

Cette version (deux-cent-dix-neuf lignes, commentaires inclus) enrichit les deux précédentes afin de pouvoir traiter n'importe quel type de *k-position* (MM_{k-p}). Nous devons entrer la taille 'D' de la grille, le nombre 'k' de régions et la dernière valeur connue de CG^*_{k-1}. Cette dernière valeur nous permet de lancer le programme en ne considérant que les MM_{k-p} «utiles», c.-à-d. ceux dont la solution sol_p correspondante vérifie : $CG(sol_p) < CG^*_{k-1}$. Cela n'accélère pas les calculs, qui doivent tous être faits (solution exhaustive), mais nous pouvons ainsi ne sauvegarder que les MM_{k-p} «utiles».

Pour traiter un nombre quelconque 'k' de régions, nous avons construit une fonction récursive (*calculRec*) du parcours des MM_{k-p} et du calcul de leur CG associé. Nous allouons dynamiquement une matrice avec autant de lignes que de région et trois colonnes. Les deux premières colonnes pour (x/y) et la troisième pour stocker, pendant le calcul récursif, la valeur du CG partiel pour le point courant du MM_{k-p} auquel il appartient. En effet nous savons que $CG(sol_p)$, correspondant à MM_{k-p}, est la somme des valeurs locales à chaque serveur S_i ($cg(sol_p,S_i)$ (*2*)) de MM_{k-p}.
Le troisième cœfficient 'z' de la matrice pour la grille (*struct ptslet xyzGrille*) permet de mémoriser, pendant le calcul récursif, le fait que le point (x/y) correspondant de la grille a déjà été traité. Cette information sera utilisée lors d'une version ultérieure du programme pour paralléliser les traitements pendant la récursivité.
Le calcul de CG(fonction *cg*) consiste, pour chaque point-motif du motif (par exemple la lettre 'A'), à trouver le point de MM_{k-p} qui lui est le plus proche. Cette distance est calculée dans le plan euclidien à deux dimensions, et non sur la grille, comme dans les premiers exemples du chapitre III. Pour optimiser les calculs nous notons dans la 3e colonne de la matrice des points-motif (*struct ptslet xyLet*) l'indice du point de MM_{k-p} le plus proche.
Une version (*calculCGstar-v2-1*) du programme a été faite sur le même principe que *calculCGstar-v2*, mais en utilisant le type *long double* dans le troisième champ de *PtsLet* au lieu du type *unsigned short*, afin de pouvoir travailler avec des réels.

CALCULCGSTAR-V3

Cette version (quatre-cent-vingt-cinq lignes, commentaires inclus) utilise la génération automatique des noms de fichier. Le fichier des motifs (lettre) s'appelle «A25-D6.data» (lettre 'A', vingt-cinq points-motif et une grille de trente-six - 6x6 - points, D=6). En sortie du programme les fichiers des multi-points s'appellent «A25-D6-R2.txt» (version non triée), «A25-D6-R2.txtT» (version triée sur CG/1re colonne) et «A25-D6-R2.txtTR» (version triée sur la valeur de la partition/dernière colonne).
Dans cette version et uniquement pour des partitions à deux régions (k=2), on calcule la valeur de la partition (*unsigned int* sur quatre octets). On utilise une fonction (*set_biti*) qui modifie la valeur des bits de la variable *unsigned int partition*. Chaque bit de la variable *partition* vaut 1 si le point-motif, correspondant à l'indice de ce bit dans *partition*, est relié (+ proche) au deuxième point du bi-points MM_{2-p}, et 0 pour le premier point de MM_{2-p}.
Cette technique ne fonctionne que pour k=2 et avec des *unsigned int* (sur quatre octets, c.-à-d. trente-deux bits). Elle ne peut représenter que des lettres d'au plus trente-deux points-lettre, l'opérateur sur les bits est limité à trente-deux. Pour généraliser cette technique à des multi-points de taille quelconque, il faut un tableau (*unsigned int partition[nbReg/2]*) où chaque élément (*partition[i]*) représente deux régions distinctes de la partition.
Il faut que toutes les régions utilisées soient différentes. Par exemple pour k=5 on aura *partition[0]* pour (Reg_1,Reg_2)

et *partition[1]* pour (Reg_3,Reg_4). La dernière région (Reg_5) n'a pas besoin d'être testée.
La gestion des fichiers de sortie (version triée,…) sera simplifiée en utilisant le shell *bash* de *Linux* et le fichier «calculCGstar-v3.sh». Il faut donner le nom du motif (A25), la dimension de la grille (D6), le *pas* de la grille ou *maille* (P1) et le nombre de régions 'k' (R3). Le nombre de points de la grille (*d_grille*) est calculé par la formule suivante: d_grille=(D/pas)-(1/pas)+1, où *pas* vaut 1 (P1) pour *maille* d'une unité, 2 (P2) pour *maille* de ½, 4 (P4) pour *maille* de ¼,…
En augmentant le *pas* de la grille (*maille*), on change la précision des calculs sans changer la position des points-motif sur la grille, ni les dimensions de cette dernière.
Si D=6, en fonction de *pas* nous avons:
- pas=1 (P1), trente-six points sur la grille;
- pas=2 (P2), cent-vingt-et-un points;
- pas=4 (P3), deux-cents-quarante-et-un points,…

Suivant la façon dont a été compilé *calculCGstar-v3* (avec ou sans *-DSANSECRIT*), le fichier des MM_{k-p} sera ou non généré.
En lançant *calculCGstar-v3* via le script, ce dernier demande d'entrer la valeur (>0) de CG^*_{k-1}. Si cette valeur est nulle, tous les MM_{k-p} seront calculés et sauvegardés, sinon seuls les MM_{k-p} «utiles» seront sauvegardés.
Uniquement dans le cas où k=2 le script génère un fichier des MM_{k-p} triés sur leur numéro de région.
La méthode qui apparaît avec cette version, consiste alors à enchaîner le script «calculCGstar-v3.sh» pour une région (R1), puis deux régions (R2) avec le $CG^*_{k=1}$,…
Pour connaître le nombre de MM_{k-p} «utiles» il suffit de compter le nombre de ligne du fichier «.txt» (*cat ADOT20-D6-P1-R1.txt|wc -l*). Comme pour les versions précédentes l'envoi du signal *SIGINT* (CTRL-C) pendant le calcul en cours permet d'obtenir son pourcentage réalisé.
Pour compiler cette version, il faut faire:

gcc calculCGstar-v3.c -o calculCGstar-v3 -Wall -ln

Une version (*calculCGstar-v1b3-1*) du programme a été faite sur le même principe que *calculCGstar-v3*, mais au lieu de fournir un nombre par partition, on fournit un nombre par région (uniquement si k=2). Pour savoir si un bi-points MM_{2-i} appartient à la même partition ($conf_1$) qu'un autre bi-points MM_{2-j} et donc en déduire tous les MM_{2-p} d'une configuration (*k-partition*) $conf_1$ donnée, il faudra comparer deux à deux chacun des nombres attribué aux régions tout en respectant l'ordre des régions (le premier avec le premier, le second avec le second,…).

Remarque: Tous ces calculs exhaustifs de $CG^*_{n,k}$, y compris ceux des version v4 et v5, sont faits en positionnant les MM_{k-p} sur la grille. On peut affiner les résultats en augmentant le *pas*, c.-à-d. en diminuant la *maille* de la grille pour augmenter le nombre de points (*d_grille*) de la grille. Les simulations que nous présenteront avec les heuristiques de résolution de $SYS_{n,k}$ sont dans l'espace euclidien réel à deux dimensions. Les résultats ainsi obtenus sont plus fins que les résultats exhaustifs qui prendraient trop de temps de calcul pour approcher la continuité des simulations. Néanmoins ces calculs exhaustifs donnent une bonne idée des résultats recherchés.

Remarque: Il existe une différence importante entre «calcul exact» et «calcul exhaustif». Le premier dépend de la précision choisie (nombre de chiffres après la virgule), dans le sens où l'exactitude l'est de plus en plus au fur et à mesure que la précision augmente; alors que le «calcul exhaustif» l'est pour une précision donnée. Autrement dit un «calcul exhaustif» peut être moins précis qu'un «calcul exact»…

CALCULCGSTAR-V4

Cette version (sept-cent-soixante-et-une lignes, commentaires inclus) distribue les calculs de la v3 sur plusieurs processus enfants (lancement de la fonction récursive *calculRec*) et utilise des IPC (six au total, cf. la figure Fig-III-C1-2) pour le partage d'informations entre eux (CG*, Mpoints*, et la grille *xyzGrille*).

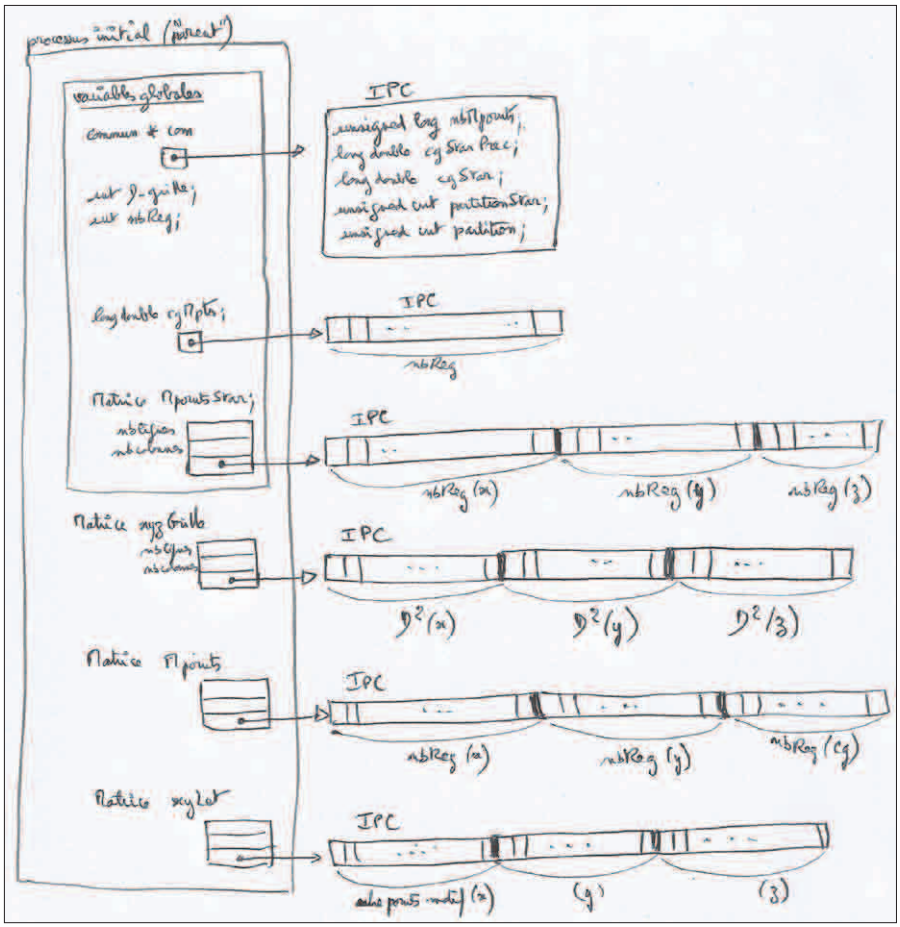


Fig-III-C1-2

L'objectif de ces déports de calculs est (v5) de profiter d'une machine multi-core (plusieurs processeurs) pour les accélérer, mais aussi (v4) de bénéficier de nouvelles piles lors de la récursivité du programme sur les différents processus enfant.
En effet la pile du processus lancé par la v3 «explose» lors de grands calculs (ADOT20 D6 P1 R11 génère plus de six-cents millions de $MM_{11\text{-}p}$ à calculer).

L'idée est de créer un processus enfant par point de la grille (D^2-2), qui lancera le calcul récursif sur les $MM_{k\text{-}p}$ correspondant à ce point.
Tous les enfants travaillent chacun leur tour (de façon séquentielle) mais sur une partie des calculs à faire.

Le processus parent attend la fin de chaque enfant (*wait*) avant de lancer le suivant, afin de ne pas avoir à synchroniser les résultats de chaque enfant, en particulier la composante 'z' des points calculés.

En retirant le *wait* dans le parent on pourrait paralléliser les calculs faits par les enfants, mais cela nécessiterait la synchronisation des valeurs partagées (cf. la version v5).

Avec les structures de données présentées sur la figure *Fig-III-C1-2*, pour manipuler les composantes (x/y/z) du point P_1 dans le code, il suffit de prendre :

xyzGrille.coeff[0]→P_1(x)

xyzGrille.coeff[0+xyzGrille.nb_lignes]→P_1(y)

xyzGrille.coeff[0+2*xyzGrille.nb_lignes]→P_1(z)

Si l'on prend une grille de trente-six points (D=6), un *pas* de 1 et k=3, on obtient quatre-vingt-quatre multi-points MM_{3-p} (7). Dans ce cas le nombre de MM_{3-p} traités par chaque processus enfant P_{E-i} est de n(n-1)/2 avec $n=D^2-i$ et $i \in [1, D^2-2]$:

P_{E-1}=7+6+5+4+3+2+1=28, soit 33%,
P_{E-2}=6+5+4+3+2+1=21, soit 25%,
P_{E-3}=5+4+3+2+1=15, soit 17,8%,
P_{E-4}=4+3+2+1=10, soit 11,9%,
P_{E-5}=3+2+1=6, soit 7,1%,
P_{E-6}=2+1=3, soit 3,57%,
P_{E-7}=1, soit 1,19%.

Plus on crée de processus enfant, plus le nombre de calculs à faire par les nouveaux processus diminue proportionnellement. Le 7e enfant n'a plus qu'un seul MM_{3-p} à calculer soit 1,19%.

Reprennons l'exemple de ADOT20-D6 (la lettre 'A' dans la police DOT, avec vingt points-motif et une grille de six (36 points). Pour k=3 nous avons 36!/(3!(36-3)!)=7140 MM_{3-p}. Les calculs se répartissent de la façon suivante sur les trente-quatre (D^2-2) processus enfant :

P_{E-1}=595, soit 8,3%,
P_{E-2}=561, soit 7,85%,
P_{E-3}=528, soit 7,39%,
…,
P_{E-9}=351, soit 4,91%,
…,
P_{E-24}=66, soit 0,92%,
…,
P_{E-33}=3, soit 0,04%,
P_{E-34}=1, soit 0,014%.

Nous ne résistons pas au plaisir de montrer deux fonctions choisies du code. La première *calculRecDeporte* crée les processus enfant :

```
void calculRecDeporte(Matrice xyz_grille, Matrice mPoints, Matrice xyLet,
          FILE* Fout, FILE* FstatProc) {
    int i_lig, status;
    int* PID; // pour mémoriser les pid des processus enfant
    PID=(int*)malloc(xyz_grille.nb_lignes*sizeof(int));
    // pour une seule région (nbReg==1) on ne lance pas de processus enfant
    // c'est un cas trivial
    if(nbReg==1) calculRec(xyz_grille, mPoints, 0, xyLet, Fout, NULL);
    else
      for (i_lig=0; i_lig<xyz_grille.nb_lignes; i_lig++) {
        PID[i_lig]=fork();
        if(PID[i_lig]==-1) perror("pb fork");
        else
          if(PID[i_lig]==0) {  // processus enfant
            // c'est le processus enfant qui fait le calcul récursif pour
            // un point de la grille; cela permet de faire D*D processus
            // enfant successifs qui ont leur propre pile.
            calculRec(xyz_grille, mPoints, i_lig, xyLet, Fout, FstatProc);
            fprintf(Fout,"%d:fin partielle du calcul déporté\n",getpid());
            exit(0);
          }
          else wait(&status); // attente de la fin de l'enfant courant
    }
    printf("%d:fin des calculs déportés\n",getpid());
}
```

La seconde *calculRec* lance le calcul récursif:

```
void calculRec(Matrice xyz_grille, Matrice mPoints, int iLig, Matrice xyLet,
          FILE* Fout, FILE* FstatProc) {
    static int profondeur=0;   // indique le P_i courant du Mpoints
    // permet de compter le nbre de MM_k-p traités par le processus enfant
    static unsigned long cptProcMP=0;
    int i_lig, j_lig, k_lig;
    long double cg_val=0;
    for (i_lig=iLig; i_lig<xyz_grille.nb_lignes; i_lig++) {
       j_lig=i_lig+1;
       // sauvegarde du point courant dans le MM_k-p
       // x du MM_k-p
       mPoints.coeff[profondeur]=xyz_grille.coeff[i_lig];
       // y du MM_k-p
       mPoints.coeff[profondeur+mPoints.nb_lignes]=
                              xyz_grille.coeff[i_lig+xyz_grille.nb_lignes];
       profondeur++;    // on positionne l'indice du MM_k-p au point suivant
       // le MM_k-p courant a été rempli, on peut lancer le calcul de CG
       if (profondeur==nbReg) {
           cg_val=cg(mPoints, xyLet);
           if (cg_val<com->cgStar) {
               com->cgStar=cg_val;     // on récupère le CG* et le MM_k-p
               for (k_lig=0; k_lig<mPoints.nb_lignes; k_lig++) {
                  MpointsStar.coeff[k_lig]=mPoints.coeff[k_lig];
                  MpointsStar.coeff[k_lig+MpointsStar.nb_lignes]=
                                mPoints.coeff[k_lig+mPoints.nb_lignes];
                  MpointsStar.coeff[k_lig+(2*MpointsStar.nb_lignes)]=
                                cgMpts[k_lig];
               }
               if(nbReg==2) com->partitionStar=com->partition;
           }
           com->nbMpoints++;  // cpteur de MM_k-p partagés par tous les processus
           cptProcMP++;       // cpteur de MM_k-p propre au processus courant
           // ce point de la grille a été traité; utilisé lors de la
           // parallèlisation du calcul pour traiter le MM_k-p suivant
           xyz_grille.coeff[i_lig+(2*xyz_grille.nb_lignes)]=1;
           profondeur--;
           …
           // raz des cg pour le MM_k-p suivant
           for (k_lig=0; k_lig<mPoints.nb_lignes; k_lig++) cgMpts[k_lig]=0;
           // si 2 régions, raz de partition ; on remet tous les bits à 0
           if(nbReg==2) memset(&com->partition,0,sizeof(unsigned int));
       }
       else {
           calculRec(xyz_grille, mPoints, j_lig, xyLet, Fout, FstatProc);
           profondeur--;
           // on arrête la récursivité pour la poursuivre sur l'enfant suivant
           if(profondeur==0) {
             // sauvegarde du nombre de MM_k-p calculés par processus, ainsi
             // que le pourcentage correspond, vis-à-vis du nbre total de MM_k-p
             if(nbReg>1)
                 fprintf(FstatProc,»%d: nbr MP=%lu\n»,getpid(), cptProcMP);
             exit(0);
           }
       }
    }
}
```

On arrête la récursivité dans un processus enfant quand profondeur=0. Dans ce cas on poursuit le calcul dans le processus enfant suivant.
Un fichier script *bash* analogue à «calculCGstar-v3.sh» a été fait pour lancer «calculCGstar-v4».

CALCULCGSTAR-V5

Cette dernière version (neuf-cent-cinquante-quatre lignes, commentaires inclus) du calcul des multi-points et de leur valeur CG(sol_p) associée, rend effective la parallélisation du code.
Cette version reprend la version «calculCGstar-v4.c» qui avait déjà créé un processus enfant par point de la grille

Processus Enfant n° D=6 / K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
2	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	
	5,56%	5,40%	5,24%	5,08%	4,92%	4,76%	4,60%	4,44%	4,29%	4,13%	3,97%	3,81%	3,65%	3,49%	3,33%	3,
3	595	561	528	496	465	435	406	378	351	325	300	276	253	231	210	
	8,33%	7,86%	7,39%	6,95%	6,51%	6,09%	5,69%	5,29%	4,92%	4,55%	4,20%	3,87%	3,54%	3,24%	2,94%	2,
4	6545	5984	5456	4960	4495	4060	3654	3276	2925	2600	2300	2024	1771	1540	1330	1
	11,11%	10,16%	9,26%	8,42%	7,63%	6,89%	6,20%	5,56%	4,97%	4,41%	3,90%	3,44%	3,01%	2,61%	2,26%	1,
5	52360	46376	40920	35960	31465	27405	23751	20475	17550	14950	12650	10626	8855	7315	5985	4
	13,89%	12,30%	10,85%	9,54%	8,35%	7,27%	6,30%	5,43%	4,66%	3,97%	3,36%	2,82%	2,35%	1,94%	1,59%	1,
6	324632	278256	237336	201376	169911	142506	118755	98280	80730	65780	53130	42504	33649	26334	20349	15
	16,67%	14,29%	12,18%	10,34%	8,72%	7,32%	6,10%	5,05%	4,14%	3,38%	2,73%	2,18%	1,73%	1,35%	1,04%	0,
7	1623160	1344904	1107568	906192	736281	593775	475020	376740	296010	230230	177100	134596	100947	74613	54264	38
	19,44%	16,11%	13,27%	10,86%	8,82%	7,11%	5,69%	4,51%	3,55%	2,76%	2,12%	1,61%	1,21%	0,89%	0,65%	0,
8	6724520	5379616	4272048	3365856	2629575	2035800	1560780	1184040	888030	657800	480700	346104	245157	170544	116280	7
	22,22%	17,78%	14,12%	11,12%	8,69%	6,73%	5,16%	3,91%	2,93%	2,17%	1,59%	1,14%	0,81%	0,56%	0,38%	0,
9	23535820	18156204	13884156	10518300	7888725	5852925	4292145	3108105	2220075	1562275	1081575	735471	490314	319770	203490	12
	25,00%	19,29%	14,75%	11,17%	8,38%	6,22%	4,56%	3,30%	2,36%	1,66%	1,15%	0,78%	0,52%	0,34%	0,22%	0,
10	70607460	52451256	38567100	28048800	20160075	14307150	10015005	6906900	4686825	3124550	2042975	1307504	817190	497420	293930	16
	27,78%	20,63%	15,17%	11,03%	7,93%	5,63%	3,94%	2,72%	1,84%	1,23%	0,80%	0,51%	0,32%	0,20%	0,12%	0,

pour lancer la fonction récursive *calculRec* sur chacun d'eux afin de soulager la pile du processus courant lors du calcul de très grands nombres de multi-points MM_{k-p}. Le programme parent attend la fin de chaque enfant (*wait*) afin de ne pas avoir à synchroniser les résultats de chaque processus.

Cette nouvelle version v5 lance en parallèle les différents processus créés ; cela nécessite de les synchroniser sur les résultats (IPC communs) pour éviter des pertes d'information ou des incohérences (recours à des sémaphores). Cette version accélère donc les calculs et permet d'aller sur des valeurs importantes de k/nbReg tout en affinant la grille et donc en améliorant les valeurs respectives des $CG^*_{n,k}$ trouvés.

Nous avons étudié l'impacte du nombre de processus enfants créés et lancés en parallèle pour le calcul des MM_{k-p} suivant l'algorithme récursif et distribué proposé dans la v5 sur l'exemple de la lettre 'A' (police DOT) sur une grille de dimension D=6 et un pas=1. Les résultats (cf. la figure *Fig-III-C1-2*) nous montrent en fonction du nombre de régions 'k', la répartition des calculs sur les trente-cinq processus enfants créés. En jaune nous avons, pour 'k' donné, 50% des MM_{k-p} traités. Si l'on ajoute le bleu d'une même ligne nous arrivons à 75% des MM_{k-p} traités.

Pour k=2 (bi-points MM_{2-p}) les onze premiers enfants traitent 50% des six-cent-trente MM_{2-p}, alors que 75% d'entre eux sont traités par les dix-huit premiers enfants sur trente-cinq.

L'étude, dont est issu le tableau de la figure *Fig-III-C1-2*, montre qu'avec k=10 nous obtenons la valeur optimale CG^*=7.7070. Ajouter des régions n'améliore pas les résultats obtenus avec k=10 à cause du déplacement des serveurs uniquement sur les points de la grille.

Donc au mieux, avec dix régions (k=10), il suffit de trois enfants pour calculer plus de 50% des 254.186.856 MM_{10-p}, et seulement cinq pour en calculer 75%.

Si l'on passe à un pas=2, c.-à-d. une grille de cent-vingt-et-un points (11*11) au lieu de trente-six, pour seulement cinq régions (k=5) il y aura 198.792.594 MM_{5-p} à calculer. Dans cette configuration le 1er processus enfant (le plus chargé) des cent-vingt prend 4,13% des MM_{5-p} à calculer. Le 50e en aura lui 0,5%.

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	nbMpoints
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		630
17%	3,02%	2,86%	2,70%	2,54%	2,38%	2,22%	2,06%	1,90%	1,75%	1,59%	1,43%	1,27%	1,11%	0,95%	0,79%	0,63%	0,48%	0,32%	0,16%		100,00%
190	171	153	136	120	105	91	78	66	55	45	36	28	21	15	10	6	3	1			7140
66%	2,39%	2,14%	1,90%	1,68%	1,47%	1,27%	1,09%	0,92%	0,77%	0,63%	0,50%	0,39%	0,29%	0,21%	0,14%	0,08%	0,04%	0,01%			100,00%
140	969	816	680	560	455	364	286	220	165	120	84	56	35	20	10	4	1				58905
94%	1,65%	1,39%	1,15%	0,95%	0,77%	0,62%	0,49%	0,37%	0,28%	0,20%	0,14%	0,10%	0,06%	0,03%	0,02%	0,01%	0,00%				100,00%
845	3876	3060	2380	1820	1365	1001	715	495	330	210	126	70	35	15	5	1					376992
29%	1,03%	0,81%	0,63%	0,48%	0,36%	0,27%	0,19%	0,13%	0,09%	0,06%	0,03%	0,02%	0,01%	0,00%	0,00%	0,00%					100,00%
5504	11628	8568	6188	4368	3003	2002	1287	792	462	252	126	56	21	6	1						1947792
80%	0,60%	0,44%	0,32%	0,22%	0,15%	0,10%	0,07%	0,04%	0,02%	0,01%	0,01%	0,00%	0,00%	0,00%	0,00%						100,00%
8760	27132	18564	12376	8008	5005	3003	1716	924	462	210	84	28	7	1							8347680
46%	0,33%	0,22%	0,15%	0,10%	0,06%	0,04%	0,02%	0,01%	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%							100,00%
7520	50388	31824	19448	11440	6435	3432	1716	792	330	120	36	8	1								30260340
26%	0,17%	0,11%	0,06%	0,04%	0,02%	0,01%	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%								100,00%
5970	75582	43758	24310	12870	6435	3003	1287	495	165	45	9	1									94143280
13%	0,08%	0,05%	0,03%	0,01%	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%									100,00%
7960	92378	48620	24310	11440	5005	2002	715	220	55	10	1										254186856
07%	0,04%	0,02%	0,01%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%										100,00%

Fig-III-C1-3

Sur une machine comme la mienne, i7 core (sept processeurs), on arrive à optimiser la répartition des calculs, mais l'ordonnanceur switche beaucoup de processus sur les processeurs avec une perte de temps indéniable. La limite de l'accélération des calculs est bien matérielle. Cette concentration des calculs sur peu de processus avec le nombre croissant de calculs à faire dépend avant tout de la nature des données traitées récursivement.

```
void calculRecDeporte(Matrice xyz_grille, Matrice xyLet, FILE* FstatProc) {
    int i_lig, status,nblignes;
    PID=(int*)malloc(xyz_grille.nb_lignes*sizeof(int));
    …
    Mpoints.coeff=(long double*)malloc(nblignes*sizeof(long double));
    …
    xyLetZ.coeff=(long double*)malloc(xyLetZ.nb_lignes*sizeof(long double));
    …
    MPSLoc.coeff=(long double*)malloc(nblignes*sizeof(long double));
    …
    // on alloue (tas) la mémoire pour les champs valeurs de ces deux structures
    partitionLocale.valeurs=(unsigned int*)malloc(partitionLocale.
                                                  nbRegions*sizeof(unsigned int));
    partitionStarLocale.valeurs=(unsigned int*)malloc(partitionStarLocale.
                                                  nbRegions*sizeof(unsigned int));
    // on met tous les bits de partitionLocale.valeurs et de
    // partitionStarLocale.valeurs à 0
    for(i_lig=0;i_lig<partitionLocale.nbRegions;i_lig++) {
      memset(&partitionLocale.valeurs[i_lig],0,sizeof(unsigned int));
      memset(&partitionStarLocale.valeurs[i_lig],0,sizeof(unsigned int));
    }
    …
    // création d'un processus enfant par point de la grille
      for (i_lig=0; i_lig<xyz_grille.nb_lignes; i_lig++) {
        PID[i_lig]=fork();
        if(PID[i_lig]==-1) perror(«pb fork»);
        else
          if(PID[i_lig]==0) {  // processus enfant
           calculRec(xyz_grille, Mpoints, i_lig, xyLet, xyLetZ, MPSLoc, f_OutPE,
                                  FstatProc, &partitionLocale, &partitionStarLocale);
           // on ne revient pas après l'appel récursif qui s'interrompt sur un
            // arrêt du processus enfant
          }
    …
    }
    for (i_lig=0; i_lig<xyz_grille.nb_lignes; i_lig++)
                                                  waitpid(PID[i_lig],&status,0);
    // on peut libérer la mémoire (tas) pour la liste des pid des enfants
    free(PID);
    free(Mpoints.coeff);
    free(xyLetZ.coeff);
    free(MPSLoc.coeff);
}
```

Pour obtenir cette parallèlisation, comme nous pouvons le voir sur le code de la fonction *calculRecDeporte* (v5) ci-dessus, on retire le *wait* dans le processus parent et on fait une boucle avec un *waitpid* sur tous les enfants à la fin de tous les calculs, dans le processus parent.

Cette version v5 nécessite donc de mettre des sémaphores pour protéger les IPCs dans lesquels les processus enfant partagent leurs résultats partiels. Cette synchronisation permet de ne pas perdre des valeurs, comme par exemple des *Mpoints* dans *nbMpoints*, dans les sections critiques de ces processus.
La figure *Fig-III-C1-4* nous montre un aperçu des structures de données manipulées par cette v5. La gestion des données critiques nous a conduit à faire évoluer cette organisation des données depuis la v4 en retirant certains IPC pour revenir à de l'allocation dynamique sur le tas (*malloc*). Nous avons introduit un tableau de sémaphores qui vont protéger des données stockées dans les IPC.

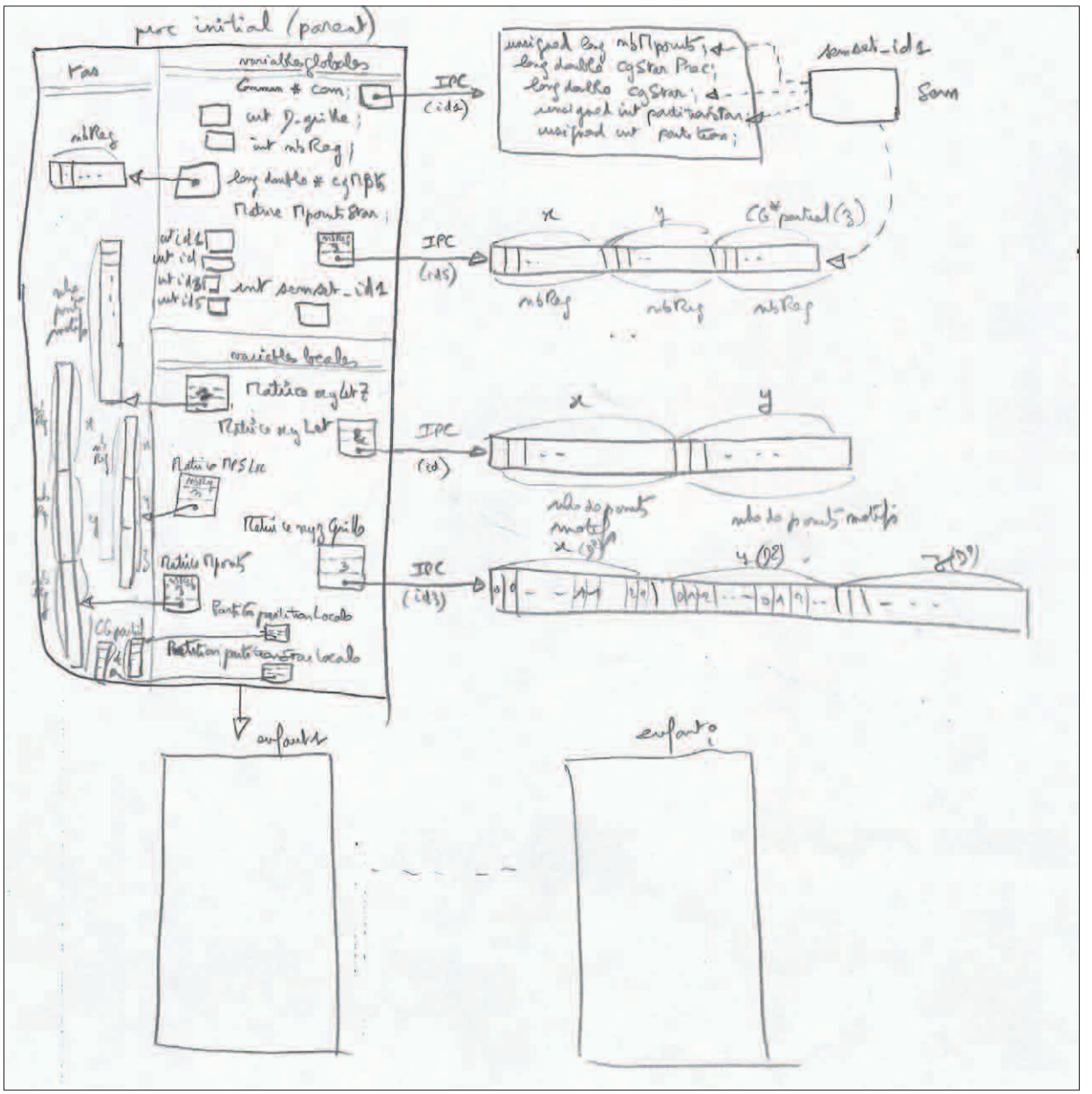


Fig-III-C1-4

Nous présentons ci-après une partie de code de la fonction *calculRec* qui gère une section critique avec les sémaphores. Il s'agit pour chaque processus enfant de mettre à jour le nombre de MM_{k-p} trouvés dans la variable partagée (IPC). En effet chaque enfant calcule ses propres MM_{k-p} en parallèle avec les autres.
Cela a été rendu possible car chacun d'eux travaille sur un point particulier de la grille. En revanche nous avons besoin de connaître le nombre total de MM_{k-p} calculés sans en perdre dans le décompte.

```
void calculRec(Matrice xyz_grille, ...) {
       …
       profondeur++;    // on positionne l'indice du MM_k-p au point suivant
       // le MM_k-p courant a été rempli, on peut lancer le calcul de CG
       if (profondeur==nbReg) {
           cg_val=cg(mPoints, xyLet, xyLetZ, partitionLocale);
           if (cg_val<cGstar) {
               …
           // ici on compte tous les MM_k-p, «utiles» ou non dans nbMpoints
           cptProcMP++;       // cpteur de MM_k-p propre au processus courant
           …
           profondeur--;

           #ifdef LISTE_Mpoints
           // écriture des MM_k-p dans un fichier; on ne garde que les MM_k-p
           // dont le CG est < au CG*_k-1; pour connaître les MM_k-p «utiles»
           // il suffira de compter les MM_k-p du fichier produit (wc -l)
           if ((Fout!=NULL)&&(cg_val<=com->cgStarPrec)) {
              …
              qsort(partitionLocale->valeurs, partitionLocale
                                    ->nbRegions,sizeof(unsigned int),cmpfunc);
              …
           #endif
            …
            else {  // on lance la récursivité
           calculRec(xyz_grille,mPoints,…);
           profondeur--;
           // c'est ici que l'on coupe la récursivité pour parallèliser les
           // calculs et la poursuivre sur l'enfant suivant
           if(profondeur==0) {
             …
////////////////////////////////////début SECTION CRITIQUE//////////////////////
            // on maj le nbMpoints commun com->nbMpoints en prenant un sémaphore
             locksem(semset_id1, 0);
             // on ajoute à la valeur commune celle qui vient d'être calculée
              // par le processus courant
             com->nbMpoints=com->nbMpoints+cptProcMP;
             // on regarde avant d'arrêter le processus enfant si son cgStar local
              // est meilleur que celui commun; si c'est le cas on sauvegarde
             if (cGstar<com->cgStar) {  // cette valeur avant de quitter le proc
                 com->cgStar=cGstar;    // ainsi que le MpointsStar local associé
                 for (k_lig=0; k_lig<mPoints.nb_lignes; k_lig++) {
                    MpointsStar.coeff[k_lig]=MPSloc.coeff[k_lig];
                    MpointsStar.coeff[k_lig+MpointsStar.nb_lignes]=
                                          MPSloc.coeff[k_lig+MPSloc.nb_lignes];
                    MpointsStar.coeff[k_lig+(2*MpointsStar.nb_lignes)]=
                                       MPSloc.coeff[k_lig+(2*MPSloc.nb_lignes)];
                 }
                 for(k_lig=0; k_lig<partitionLocale->nbRegions; k_lig++)
                   com->partitionStar.valeurs[k_lig]=
                                          partitionStarLocale->valeurs[k_lig];
             }
             // on libère le sémaphore
             unlocksem(semset_id1, 0);
////////////////////////////////////fin SECTION CRITIQUE////////////////////////////
             …
             free(partitionStarLocale->valeurs);
             // fin du processus enfant courant
             exit(0);
             …
}
```

Pour la suite des traitements nous avons besoin de trouver les MM_{k-p} par configuration $conf_1$, en particulier nous nous intéressons à la *k-partition* optimale $conf^*_1$. Pour cela il faut calculer un nombre entier unique par configuration $conf_1$.

Nous proposons une méthode efficace en temps de calcul et en utilisation d'espace mémoire, qui fonctionne jusqu'à des motifs ayant trente-deux points-motif au plus, ce qui implique un nombre de régions (k) également limité à trente-deux. Ceci va être suffisant pour notre étude.

Nous créons (cf. la figure *Fig-III-C1-5*) une structure *Partition partitionLocale* sur le tas, dont le troisième champ *unsigned int* valeur* est un tableau d'entier, alloué aussi sur le tas. Chaque entier de ce tableau correspond à l'une des régions de la partition courante $conf_1$. Il est codé sur trente-deux bits représentant les trente-deux points-motif du motif.

Le premier nombre ([0]) correspond au premier point P_1 de MM_{k-p}, le second à P_2,… Tous les bits du même rang pour les trente-deux nombres sont exclusifs entre eux. Un seul d'entre eux vaudra 1.

Cette technique permet un post-traitement simple et efficace des données. En effet en faisant *grep* sur le fichier des MM_{k-p}, avec le numéro d'une *k-partition*, on obtient tous les MM_{k-p} de cette *k-partition* et leur nombre.

Voici un exemple complet représenté sur la figure *Fig-III-C1-6* avec un MM_{3-p} et un motif composé de cinq points-motif (PM_i).

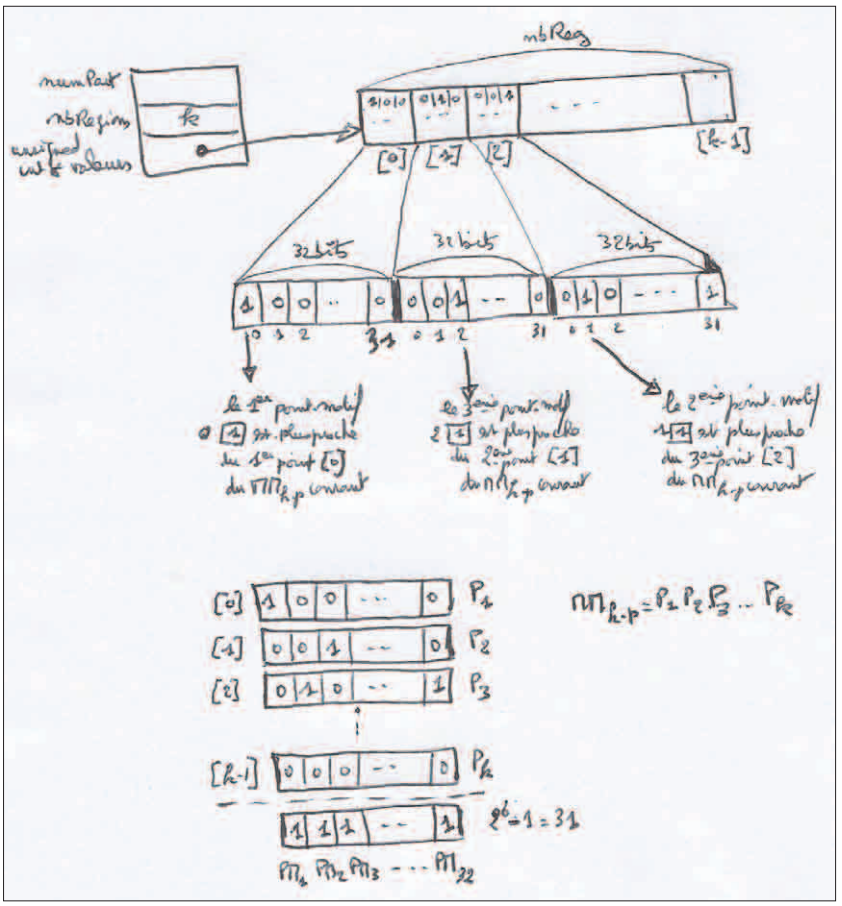


Fig-III-C1-5

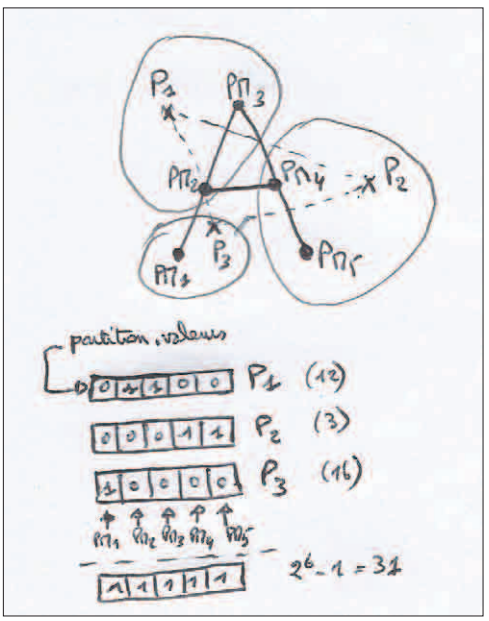


Fig-III-C1-6

Pour lancer *calculCGstar-v5* nous avons construit le fichier script (bash) «calculCGstar-v5.sh». Ce dernier demande le nom du motif, la dimension de la grille, le *pas* de la grille et le nombre de régions. Par exemple: ADOT20 D6 P2 R2, avec le fichier «ADOT20-D6.data» pour les coordonnées des points-motif.

Le script demande si l'on souhaite travailler avec tous les points de la grille ou uniquement ceux qui englobent l'enveloppe convexe du motif (points «utiles»). Suivant les motifs étudiés il peut être très intéressant de ne considérer que les points de la grille où il y a une chance que les points P_i du MM_{k-p} courant puissent se positionner utilement.

En effet une grande partie de la complexité en temps de calcul provient du nombre de points de la grille. Nous avons d'ailleurs vu que plus ce nombre est grand et moins la répartition des calculs sur les différents processeurs est homogène.
De plus c'est assez simple de construire une bonne fois pour toute (dans le fichier «xyzGrilleMin.data») la liste de ces points «utiles» de la grille. Ils serviront pour toutes les expérimentations. Dans notre cas les points du MM_{k-p} sont forcément dans l'enveloppe convexe du motif, c'est la raison pour laquelle en procédant de la sorte on ne risque pas d'oublier un point intéressant.

```
#! /bin/bash
# en entrée : A25 D6 P1 R2 (motif, grille, pas de grille et nbre de régions)
# en sortie : A25-D6-P1-R2.txt (ens. des MM_k-p utiles non triés)
#             A25-D6-P1-R2.txtT (ens. des MM_k-p utiles triés par CG)
#             A25-D6-P1-R2.txtP (nbre de MM_k-p calculés par processus enfant)
#             Si R2 (deux régions par partition) uniquement
#             A25-D6-P1-R2.txtTR (les MM_k-p «utiles» triés par partition et par
#              CG croissant dans chaque partition - uniquement avec deux régions)
# pour ne pas récupérer les MM_k-p, mais uniquement afficher CG* et MP* en fin de
# calcul, il suffit de compiler «calculCGstar-v5.c» avec l'option -DSANSECRIT
echo «entrez : motif grille pasGrille regions»
echo «par exemple : A25 D6 P1 R2 (pasGrille : P1 ou P2 ou P4 ou P8 ou P16…)»
read motif grille pasGrille regions
echo «voulez-vous faire un calcul avec uniquement les points utiles de la grille (o/n) ?»
read reponse
if [ $reponse = o ]
then
  if test -f $motif-$grille-$pasGrille.dataMin
  then
    cp $motif-$grille-$pasGrille.dataMin xyGrilleMin.data
    echo «le calcul va être fait sur les points utiles de la grille»
  else
    echo «il manque le fichier des points utiles de la grille Motif-Grille-Pas.dataMin»
    exit 0
  fi
else
  rm xyGrilleMin.data 2> /dev/null
  echo «le calcul va être fait sur la grille entière»
fi
################################################################################
# lancement du calcul des MM_k-p
calculCGstar-v5 $motif $grille $pasGrille $regions
################################################################################
# dans la mesure où plusieurs processus enfants vont calculer en parallèle leurs
# MM_k-p respectifs, chacun d'eux va produire un fichier «.txtN°pid». Cela va
# permettre d'éviter de mettre un sémaphore sur l'écriture dans le fichier «.txt»
# qui aurait ralenti tous les enfants
# puis concaténation des fichiers produits dans le fichier «.txt» qui sera trié
if test -f $motif-$grille-$pasGrille-$regions.txt
then
  # avant de trier le fichier, il faut le remplir avec les différents fichiers
  # de MM_k-p remplis par les différents processus enfants
  for i in $motif-$grille-$pasGrille-$regions.txt[0-9]*
  do
    cat $i >> $motif-$grille-$pasGrille-$regions.txt
    rm $i
  done
  # il faut trier le fichier (ex. «A25-D6-P1-R2.txt») des MM_k-p issu de
  # calculCGstar-v5 et mettre le résultat dans le fichier «.txtT» correspondant
  sort -n $motif-$grille-$pasGrille-$regions.txt > $motif-$grille-$pasGrille-$regions.txtT
  echo «nbre de MM_k-p utiles :»
  cat $motif-$grille-$pasGrille-$regions.txt | wc -l
  # FONCTIONNE UNIQUEMENT AVEC 2 REGIONS
  if [ $regions = R2 ]
  then
    sort -g -k 9 $motif-$grille-$pasGrille-$regions.txt > $motif-$grille-
$pasGrille-$regions.txtTR
  fi
else
  echo «pas de création de fichiers pour les MM_k-p»
fi
echo «Fin du calcul»
```

calculGCstar-v5.sh

Attention, si l'on choisi cette approche avec uniquement les points «utiles» de la grille, le *pas* n'est plus pris en compte, même s'il est défini.
Pour des questions de lisibilité des expérimentations nous adopterons la règle suivante :

- «ADOT20-D6.data» → le motif 'A' avec vingt PM_j sur la grille complète ;
- «ADOT20-D6-P2-42pts.dataMin» → idem mais sur un *pas* de 2 (cent-vingt-et-un points) et quarante-deux points sur la grille ;
- «ADOT20-D6-P2-30pts.dataMin» → idem mais sur un *pas* de 2 (cent-vingt-et-un points) et trente points sur la grille.

Une technique consiste à faire des calculs peu coûteux sur cent-vingt-et-un points (par exemple k=2) pour trouver de façon la plus précise possible l'ensemble des points «utiles» de la grille. Ensuite on poursuit les calculs avec des valeurs de 'k' supérieures uniquement sur ces points «utiles».
Par contre lorsque l'on travaillera sur la recherche de $CG^*_{n,k}$, c.-à-d. sur la recherche du MM^*_{k-p} qui donne la configuration optimale $conf^*_1$, il pourra être intéressant d'agrandir la grille pour capter davantage de MM_{k-p} et ainsi mieux voir le champ des MM_{k-p} de $conf^*_1$.
Le script nous propose donc deux scénarii possibles :

1. toute la grille ou une partie (les points «utiles») ;
2. tous les MM_{k-p} ou uniquement les MM_{k-p} «utiles.

GenVector-v2.sh

Cet outil nous permet, à partir des MM_{k-p} (par exemple «A25-D6-P2-R2.**txt**») issus du programme précédent, de produire les données suivantes :

- «A25-D6-P2-R2.**txtT**» : les MM_{k-p} triés sur la 1re colonne (CG) ;
- «A25-D6-P2-R2.**txtTR**» : les MM_{k-p} triés par partition, sur les 'k' colonnes, vues comme une seule et par ordre croissant sur CG dans chaque partition ;
- «A25-D6-P2-R2.**txtTRnbP**» : le nombre de partitions, utile pour l'adjacence des MM_{k-p} des partitions ;
- «A25-D6-P2-R2.**txtTRRP**» : les régions (int) par partition ;
- «A25-D6-P2-R2.**txtTRPM**» : les PM_i par région et partition ;
- «A25-D6-P2-R2.**txtTRS**» : uniquement les MM_{k-p} de CG^* ;
- «A25-D6-P2-R2.**txtTRSV**» : les vecteurs (x1 y1 dx dy) des Mpoints de CG^* ;
- «A25-D6-P2-R2.**txtTRV**» : les vecteurs des MM_{k-p} par partition.

Nous avons consacré cinq versions successives pour obtenir le programme en langage 'C' («calculCGstar-v5.c») qui calcule tous les MM_{k-p} d'un problème donné $SYS_{n,k}$. Nous avons même introduit un script *shell* pour accompagner son lancement.
Il est courant en informatique, et tout particulièrement en programmation système, mais aussi lorsque vous traitez des données comme dans cette étude, de combiner des outils différents pour aller au plus vite vers la solution recherchée.
Ceci nous conduit naturellement à construire le script *shell* «GenVector-v2.sh» qui nous intéresse pour produire les fichiers résultats énumérés ci-dessus.

Cette imbrication est telle que «GenVector-v2.sh» appelle lui-même un programme écrit en 'C', «calculregionsPts.c», pour le calcul du fichier «.txtTRPM» des points-motif (PM_i) par région (Reg_j) et par partition ($conf_l$).
Le script va aussi utiliser, entre autres, les outils très puissants que sont *awk* et *sed*.
Voici un extrait du code de ce script :
À ce stade du script nous avons dans («.txtTR») tous nos MM_{k-p} triés sur les 'k' colonnes correspondant aux numéros de chaque région (Reg_i), et considérées comme une seule colonne. Nous avons aussi («.txtTRRP») les régions par partition ($conf_l$). Nous avons enfin («.txtTRPM») les points-motif (PM_i) par région et par partition.

```
#################################################################################################
# calcul de «.txtTR», de «.txtTRRP» et de «.txtTRPM»
#################################################################################################
# on trie «.txt» en «.txtTR», sur les nbReg dernières colonnes, vues comme une seule
rm ${d1}TR 2> /dev/null # on le supprime pour ne pas concaténer des MMk-p avec des anciens
rm ${d1}TRRP 2> /dev/null # on le supprime pour ne pas concaténer avec des anciennes valeurs
rm ${d1}TRPM 2> /dev/null # on le supprime pour ne pas concaténer avec des anciennes valeurs
nbMP=$(cat ${d1}T|wc -l) # on récupère le nbre de lignes, donc de MMk-p de «.txtT»
cp ${d1}T fichTR.txt      # on recopie «.txtT» dans le fichier de travail «fichTR.txt»
cptPart=0                 # nbre de partitions trouvées
while [ $nbMP -gt 0 ]     # pour chaque MMk-p du fichier de travail
 do
  i=0
  while [ $i -lt $nbReg ] ; do # on récupère les «nbReg» identifiants (valeur) des régions
                               # de la partition à laquelle appartient le MMk-p courant
    pStar[i]=$(sed '2,$d' fichTR.txt|awk -v numPremReg=$numPremReg '{print $numPremReg }')
                               # sed permet de ne garder que la première ligne du fichier
                               # correspondant au MMk-p
    # on stocke dans «.txtTRRP» la valeur de la région en entier
    echo « ${pStar[i]}» >> ${d1}TRRP
    # pStar contient à la fin les différentes valeurs correspondant aux régions
    var=$var» ${pStar[i]}»     # on concatène dans var les diff valeurs (nbReg) des régions de P
    numPremReg=$(($numPremReg+1))    # on passe au nbre suivant
    i=$(($i+1))
  done
  grep «$var» fichTR.txt | wc -l >> ${d1}TR   # on sauvegarde ce nbre dans «.txtTR», il
                                # sera utilisé pour calculer les MMk-p adjacents
  grep «$var» fichTR.txt >> ${d1}TR  # on sauvegarde dans «.txtTR» les MMk-p de la partition courante
  echo  «FINPARTITION» >> ${d1}TR # entre chaque partition pour gnuplot (bloc, mot-cle:index)
  grep -v «$var» fichTR.txt > fichTRbis.txt # on retire les MMk-p de la partition courante («-v»)
  cp fichTRbis.txt fichTR.txt     # on recopie le résultat sur le fichier «fichTR.txt»
  nbMP=$(cat fichTR.txt|wc -l)  # on maj le nbre de MMk-p à traiter dans le fichier «fichTR.txt»
  cptPart=$(($cptPart+1))       # on incrémente le nbre de partitions trouvées
  numPremReg=$(($3*3+3))
  echo « « >> ${d1}TRRP
  unset var
done
# construction du fichier «.txtTRPM» des points-motif par région et par partition
calculRegionsPts $1 $d1 $nbReg $cptPart >> ${d1}TRPM
echo «Nbre de partitions trouvées=$cptPart»
echo «$cptPart» > ${d1}TRnbP
rm fichTR.txt 2> /dev/null
rm fichTRbis.txt 2> /dev/null
#################################################################################################
```

GenVector-v2.sh (extrait)

Pour étudier la partition optimale ($conf^*_l$) nous construisons le fichier («.txtTRS») qui contient tous les MM_{k-p} de $conf^*_l$.
Dans la partie du script qui effectue ce traitement on voit l'imbrication fine qu'il peut y avoir entre le script écrit en shell *bash* et la commande *awk*.
En effet on va récupérer par *awk* un contenu dans sa variable *numPremReg*. Ce contenu va être transmis à la variable homonyme du shell *numPremReg*, pour la suite du script.
La suite du script calcule le fichier («.txtTRSV») qui permet, dans *gnuplot*, l'affichage des vecteurs correspondant

aux MM_{k-p} de $conf^*_1$. Pour cela on présente le fichier sous la forme de quatre colonnes (x1 y1 dx dy).
Enfin on construit le fichier («.txtTRV») qui contient tous les MM_{k-p} de toutes les partitions de $SYS_{n,k}$, toujours sous

```
##########################################################################################
# calcul de «.txtTRS»
##########################################################################################
# écriture des lignes de la partition* (conf*1) dans le fichier «.txtTRS»
# ce fichier permet dans gnuplot d'afficher les puits des MMk-p de conf*1
i=0
while [ $i -lt $nbReg ] ; do  # on lance des sed/awk successifs pour positionner dans
                              # var les différentes valeurs des régions de conf*1
pStar[i]=$(sed '2,$d' ${d1}T|awk -v numPremReg=$numPremReg '{print $numPremReg }')
                              # nbre courant; sed permet de ne garder que la première
                              # ligne du fichier correspondant a CG*
                              # on ne garde que (suppression des lignes 2 a $ - dernière
                              # ligne) la 1re ligne de «.txtT»; le fichier «.txtT» est
                              # inchangé; awk affiche le nbre de cette ligne
                              # correspondant à la colonne dans la variable
                              # numPremReg; on note ici le passage de la variable
                              # numPremReg du shell a celle d'awk
                              # echo ${pStar[i]}, pour afficher la ie case du tableau pStar[]
                              # pStar contient, à la fin, les différentes valeurs de P* (CG*)
var=$var» ${pStar[i]}»        # on concatène dans var les diff valeurs (nbReg) des régions de CG*
numPremReg=$(($numPremReg+1)) # on passe au nbre suivant
i=$(($i+1))
done
grep «$var» ${d1}T > ${d1}TRS # faire un grep de var dans tout le fichier «.txtT»
##########################################################################################
```

GenVector-v2.sh (extrait)

la forme d'un quadruplet (x1 y1 dx dy). Ce fichier est aussi utilisé par *gnuplot* pour visualiser les MM_{k-p}.
Sans montrer le code de cette partie du script il est intéressant de signaler que nous aurions pu faire ces calculs dans un script *awk* avec ses propres paramètres positionnels. Nous avons préféré rester dans le script *bash* en utilisant la commande *shift* (décalage des paramètres positionnels du *bash*) afin de rester sur la même ligne du script.

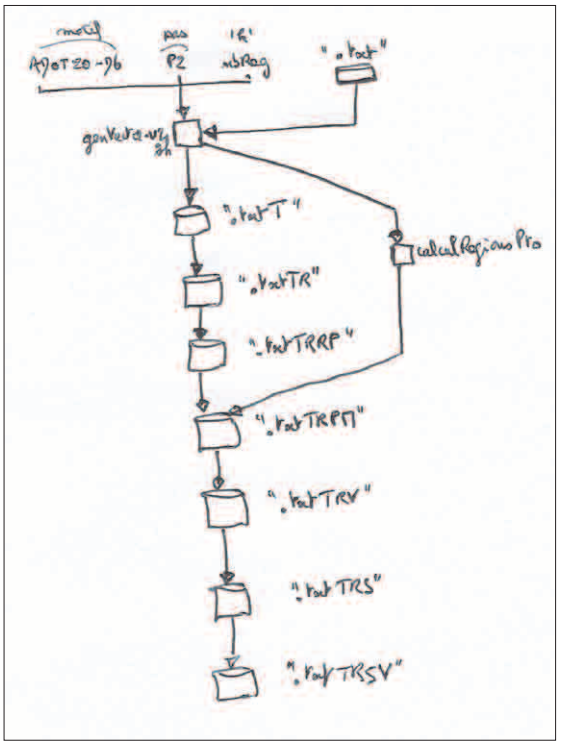


Fig-III-C1-7

En résumé de ces premiers outils, voici ce qu'il faut lancer :

1. «calculCGstar-v5.sh» (-> ADOT20 D6 P2 R3, o/n, CG^*_{k-1})
2. «genVector-v2.sh ADOT20-D6-P2-R3.txt nbReg»
3. «gnuplot» (-> load «config_gnuplot.gnu»
-> plot «ADOT20-D6-P2-R3.txtTRSV» u 1:2:3:4 with vectors nohead // pour $conf^*_1$
-> plot «ADOT20-D6-P2-R3.txtTRV» index i u 1:2:3:4 with vectors nohead // pour
$conf_{i-1}$(0 <=> $conf^*_1$)

Pour illustrer ces outils nous allons voir quelques résultats que l'on peut obtenir en partant de l'exemple du motif 'A' (police DOT) avec vingt points-motif (PM_i) sur une grille de trente-six points (D=6) avec une maille de ½ (pas=2), soit cent-vingt-et-un points au maximum et deux régions (k=2). Nous ne garderons que les trente points «utiles» de la grille qui recouvrent l'enveloppe convexe du motif. Ces contraintes se traduisent par deux fichiers : «ADOT20-D6.data» (PM_i) et «ADOT20-D6-P2-30pts.dataMin» (les points «utiles» de la grille).

Le fichier «ADOT20-D6-P2-R2.txtTR» comprend les $MM_{2\text{-}p}$ par partition en partant de $conf^{*}_{1}$ et jusqu'à la «moins bonne» (celle dont le CG_{min} est le plus grand de tous les CG_{min} des autres partitions de $SYS_{n,k}$) :

```
22
18.740960271699346 2.000000 1.500000 14.254167223859142 2.000000 3.500000 4.486793047840205 4 4064 1044511
18.740960271699346 2.000000 1.500000 14.254167223859142 2.500000 3.500000 4.486793047840205 12 4064 1044511
18.824706261607715 2.000000 3.500000 4.486793047840205 2.500000 1.500000 14.337913213767510 4 4064 1044511
18.824706261607715 2.500000 1.500000 14.337913213767510 2.500000 3.500000 4.486793047840205 4 4064 1044511
19.257415311856109 2.000000 1.500000 14.254167223859142 2.000000 3.000000 5.003248087996967 3 4064 1044511
19.341161301764477 2.500000 1.500000 14.337913213767510 2.500000 3.000000 5.003248087996967 3 4064 1044511
19.351374276238246 2.000000 1.000000 14.864581228398041 2.000000 3.500000 4.486793047840205 5 4064 1044511
19.425357507585304 2.500000 1.000000 14.938564459745099 2.500000 3.500000 4.486793047840205 5 4064 1044511
20.745330735668749 2.000000 1.000000 14.864581228398041 2.000000 4.000000 5.880749507270708 6 4064 1044511
20.745330735668749 2.000000 1.000000 14.864581228398041 2.500000 4.000000 5.880749507270708 14 4064 1044511
20.819313967015807 2.000000 4.000000 5.880749507270708 2.500000 1.000000 14.938564459745099 2 4064 1044511
20.819313967015807 2.500000 1.000000 14.938564459745099 2.500000 4.000000 5.880749507270708 6 4064 1044511
21.504157630421071 2.000000 2.000000 16.500909542424104 2.000000 3.000000 5.003248087996967 2 4064 1044511
21.554798666544672 2.500000 2.000000 16.551550578547705 2.500000 3.000000 5.003248087996967 2 4064 1044511
22.064410749787656 1.500000 1.000000 16.183661242516948 2.000000 4.000000 5.880749507270708 11 4064 1044511
22.231144006835749 2.500000 4.000000 5.880749507270708 3.000000 1.000000 16.350394499565041 2 4064 1044511
23.599848305130563 2.000000 2.000000 16.500909542424104 2.000000 2.500000 7.098938762706459 1 4064 1044511
23.650489341254163 2.500000 2.000000 16.551550578547705 2.500000 2.500000 7.098938762706459 1 4064 1044511
23.783935938120654 2.000000 0.500000 17.903186430849946 2.000000 4.000000 5.880749507270708 7 4064 1044511
23.830794005326378 2.500000 0.500000 17.950044498055671 2.500000 4.000000 5.880749507270708 7 4064 1044511
26.177482064114119 1.500000 2.000000 17.770815103024958 1.500000 2.500000 8.406666961089161 1 4064 1044511
26.306875414041866 3.000000 2.000000 17.900208452952705 3.000000 2.500000 8.406666961089161 1 4064 1044511
FINPARTITION
15
18.980157863488399 2.000000 1.500000 13.033511662285771 2.500000 3.000000 5.946646201202628 11 8160 1040415
19.182163168775327 2.000000 1.000000 13.302531293216710 2.500000 3.500000 5.879631875558617 13 8160 1040415
19.817947105742458 1.500000 1.500000 13.938315230183841 2.500000 3.500000 5.879631875558617 17 8160 1040415
20.065230224709719 1.500000 1.500000 13.938315230183841 2.000000 3.500000 6.126914994525878 9 8160 1040415
20.142355846845710 1.500000 1.000000 14.262723971287094 2.500000 3.500000 5.879631875558617 18 8160 1040415
20.222188165667379 1.500000 1.500000 13.938315230183841 2.000000 3.000000 6.283872935483537 8 8160 1040415
20.389638965812971 1.500000 1.000000 14.262723971287094 2.000000 3.500000 6.126914994525878 10 8160 1040415
22.011627647784742 1.500000 1.000000 14.262723971287094 2.500000 4.000000 7.748903676497648 19 8160 1040415
22.184347648152408 2.500000 1.500000 13.477680687063247 3.000000 2.500000 8.706666961089161 10 8160 1040415
22.541413443466339 1.500000 2.000000 16.257540507982802 2.000000 3.000000 6.283872935483537 7 8160 1040415
23.679781815015993 2.000000 0.500000 15.930878138518344 2.500000 4.000000 7.748903676497648 15 8160 1040415
24.588587814407337 1.500000 0.500000 16.839684137909688 2.500000 4.000000 7.748903676497648 20 8160 1040415
24.654108334362236 1.000000 1.000000 16.905204657864588 2.500000 4.000000 7.748903676497648 21 8160 1040415
24.779559673377796 1.500000 0.500000 16.839684137909688 2.000000 4.000000 7.939875535468108 12 8160 1040415
24.845080193332696 1.000000 1.000000 16.905204657864588 2.000000 4.000000 7.939875535468108 13 8160 1040415
FINPARTITION
15
19.063903853396767 2.000000 3.000000 5.946646201202628 2.500000 1.500000 13.117257652194139 5 4080 1044495
19.256146400122385 2.000000 3.500000 5.879631875558617 2.500000 1.000000 13.376514524563768 3 4080 1044495
19.996839717011585 2.000000 3.500000 5.879631875558617 3.000000 1.500000 14.117207841452969 12 4080 1044495
20.244122835978846 2.500000 3.500000 6.126914994525878 3.000000 1.500000 14.117207841452969 4 4080 1044495
20.309089103893803 2.000000 3.500000 5.879631875558617 3.000000 1.000000 14.429457228335186 11 4080 1044495
20.401080776936506 2.500000 3.000000 6.283872935483537 3.000000 1.500000 14.117207841452969 5 4080 1044495
20.556372222861064 2.500000 3.500000 6.126914994525878 3.000000 1.000000 14.429457228335186 3 4080 1044495
22.100601658244040 1.500000 2.500000 8.706666961089161 2.000000 1.500000 13.393934697154879 3 4080 1044495
22.178360904832835 2.000000 4.000000 7.748903676497648 3.000000 1.000000 14.429457228335186 10 4080 1044495
22.670806793394087 2.500000 3.000000 6.283872935483537 3.000000 2.000000 16.386933857910550 6 4080 1044495
23.726639882221717 2.000000 4.000000 7.748903676497648 2.500000 0.500000 15.977736205724069 1 4080 1044495
24.710425908751042 2.000000 4.000000 7.748903676497648 3.000000 0.500000 16.961522232253393 9 4080 1044495
24.846367032821810 2.000000 4.000000 7.748903676497648 3.500000 1.000000 17.097463356324162 15 4080 1044495
24.901397767721501 2.500000 4.000000 7.939875535468108 3.000000 0.500000 16.961522232253393 1 4080 1044495
25.037338891792269 2.500000 4.000000 7.939875535468108 3.500000 1.000000 17.097463356324162 7 4080 1044495
FINPARTITION
…
```

Nous avons ci-dessus les trois premières partitions en partant de $conf^{*}_{1}$ (avec vingt-deux $MM_{2\text{-}p}$), puis $conf_{1}$ (avec quinze $MM_{2\text{-}p}$)… On voit les $MM_{2\text{-}p}$ rangés par CG (1re colonne) croissant à l'intérieur de chaque partition. Les deux dernières colonnes correspondent aux nombres entiers identifiant de façon unique chaque région (deux régions donc deux nombres) et par conséquence chaque partition $conf_{1}$.

Pour afficher dans *gnuplot* uniquement les $MM_{2\text{-}p}$ «utiles» d'une partition $conf_{1}$ donnée (cf. les figures *Fig-III-C1-8/9/10*) nous avons construit un fichier par partition appelé «ADOT20-D6-P2-R2.txtTRMPstar» pour $conf^{*}_{1}$, ou encore «ADOT20-D6-P2-R2.txtTRMP1» pour la seconde,…

La figure *Fig-III-C1-8* représente les $CG(sol_{p})$ des différentes solutions sol_{p} des $MM_{2\text{-}p}$ de la configuration optimale $conf^{*}_{1}$, la figure *Fig-III-C1-9* ceux de la 2e

configuration et la figure *Fig-III-C1-10* ceux de la 3e.
Enfin la figure *Fig-III-C1-11* cumule toutes les solutions de toutes les configurations.

Nous allons pouvoir étudier la part locale (cg) de chaque région (Reg_i) d'une partition $conf_l$ donnée, dans le coût global (CG) de la solution obtenue.
La figure *Fig-III-C1-12* nous montre avec les deux courbes basses, les parts locales des deux régions de la configuration optimale $conf^*_l$. La courbe haute est le coût de la solution globale (CG^*_{2-k}).
Les figures *Fig-III-C1-13/14/15/16* montrent les MM_{2-p} «utiles» pour le motif 'A'.

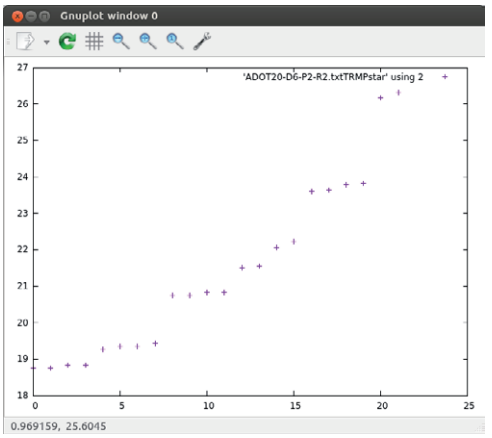


Fig-III-C1-8

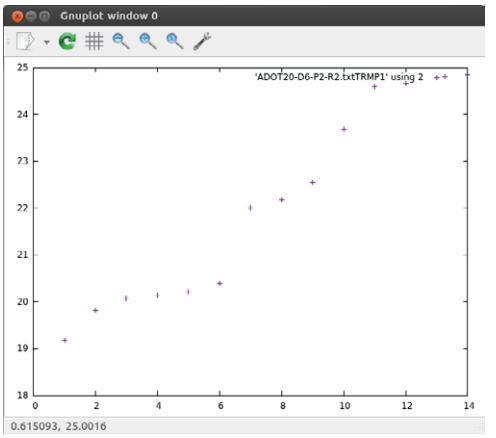


Fig-III-C1-9

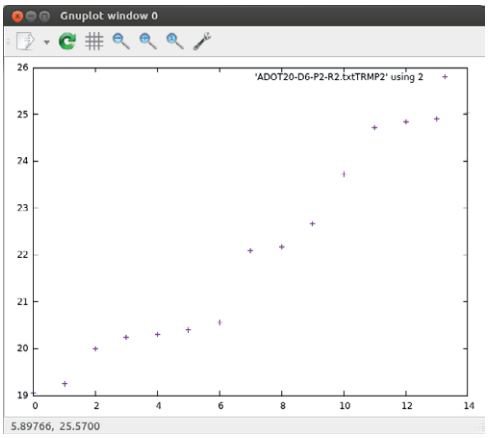


Fig-III-C1-10

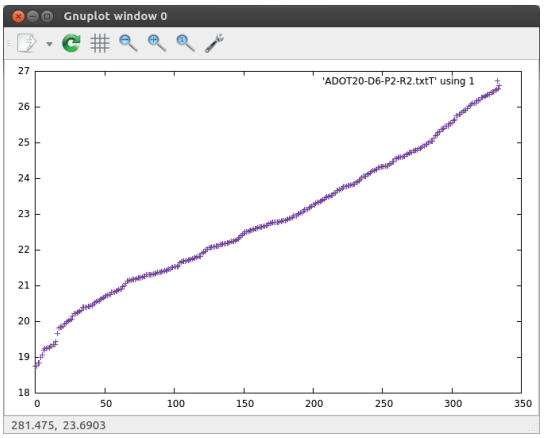


Fig-III-C1-11

La figure *Fig-III-C1-13* représente les MM_{2-p} «utiles» de la partition $conf^*_l$ optimale; la figure *Fig-III-C1-14* ceux de la 2e configuration et la figure *Fig-III-C1-15* ceux de la 3e. Enfin la figure *Fig-III-C1-16* superpose ceux de toutes les configurations du motif 'A'.

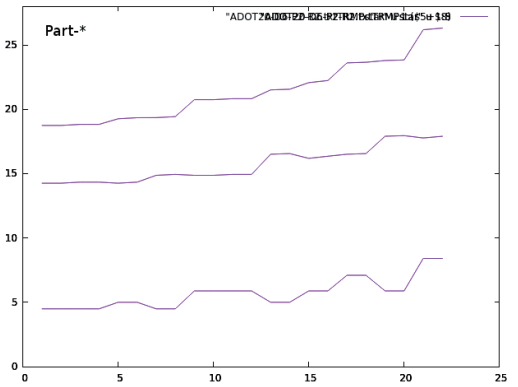


Fig-III-C1-12

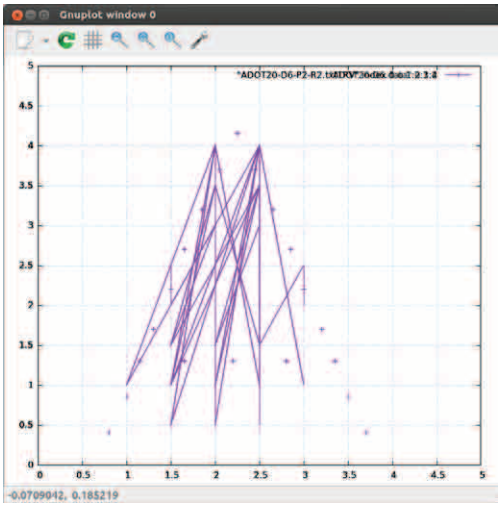


Fig-III-C1-13

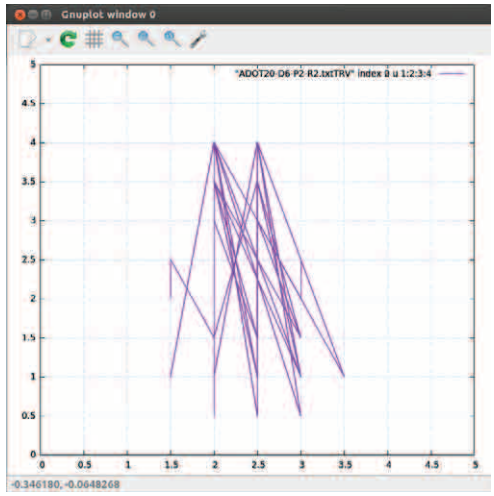


Fig-III-C1-14

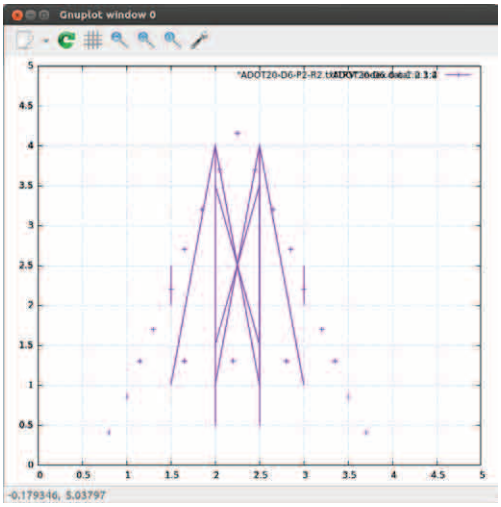


Fig-III-C1-15

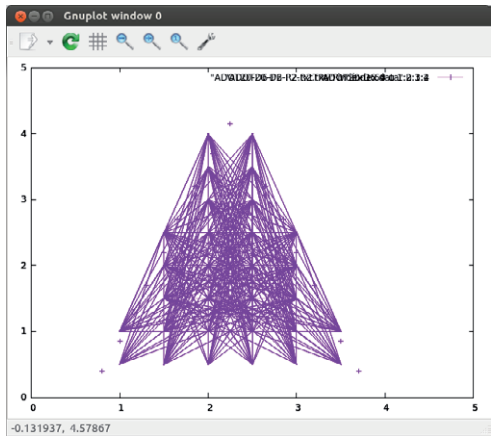


Fig-III-C1-16

III-C2) CALCUL DES GRAPHES D'ADJACENCE DES *K-PARTITIONS*

Il s'agit de construire les graphes d'adjacence ou de voisinage entre les *k-partitions* $conf_1$ afin de comprendre comment les heuristiques exploratoires de résolution pourront fonctionner efficacement.
Trois outils, écrits en langage 'C', sont proposés. Ils vont tous les trois produire des graphes dans le langage compréhensible par le générateur de graphes *GraphStream*.

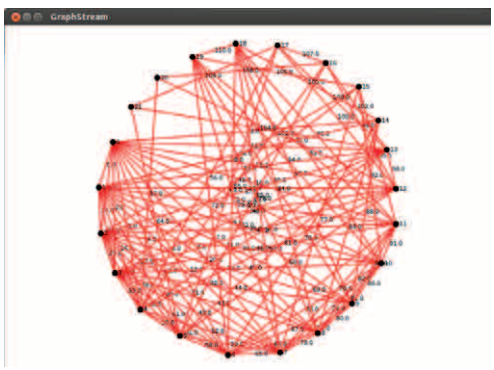


Fig-III-C2-17

ADJACENCE ENTRE LES MM_{k-p} D'UNE MÊME *K-PARTITION* $CONF_1$

Le premier outil «calculAdjacenceMP.c» calcule toutes les adjacences internes, c.-à-d. entre tous les MM_{k-p} d'une *k-partition* $conf_1$ donnée. Il prend en entrée les fichiers «.txtTR» (ensemble des MM_{k-p} par *k-partition* $conf_1$) et «.txtTRnbP» (nombre de *k-partitions*). Les résultats sont stockés dans le fichier «.txtTRdgs». Chaque partition a son propre graphe (sur la figure *Fig-III-C2-17* on voit le graphe de $conf^*_1$). Il existe également le graphe des MM_{k-p} inter *k-partitions*, c.-à-d. reliant les *k-partitions* entre elles.

Conformément aux définitions vues dans la partie III-B3 du chapitre III (cf. la figure *Fig-III-B3-17*) deux multi-points MM_{k-1} et MM_{k-2} sont *q-adjacents* l'un l'autre, si tous les points P_i de MM_{k-1} sont dans le *q-voisinage* d'un et d'un seul point P_j de MM_{k-2} (et réciproquement). Sur la figure *Fig-III-C2-18*, cela se traduit par:
$P_i(x_i,y_i)$ est dans le *8-voisinage* de $P_j(x_j,y_j)$ $\Leftrightarrow$ $x_i \in [x_j-d, x_j+d]$ et $y_j \in [y_j-d, y_j+d]$.

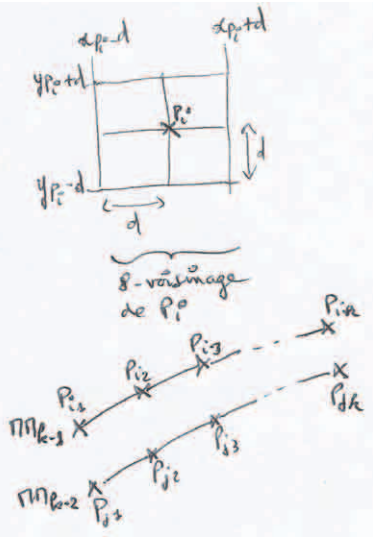


Fig-III-C2-18

Pour tester ce voisinage de façon efficace nous utilisons deux vecteurs V_1 et V_2 de 'k' cases toutes initialisées à 0. Nous pourrions utiliser directement les bits d'une variable entière. Le gain de place n'en vaut pas la peine ici. Si P_i est dans le *q-voisinage* de P_j alors $V_1[i]=V_2[j]=1$. Si à la fin l'une des cases de V_1 ou de V_2 est à 0 cela signifie que les deux multi-points MM_{k-1} et MM_{k-2} ne sont pas adjacents. En effet l'adjacence de deux multi-points nécessite le passage de l'un à l'autre en un seul déplacement par point.

ADJACENCE ENTRE LES *K-PARTITIONS* $CONF_L$

Le second programme «calculAdjacencePart.c» calcule, en plus du précédent, l'adjacence des *k-partitions* entre elles.
Sur la figure *Fig-III-C2-19* nous avons trois représentations différentes du même graphe des adjacences entre les différentes partitions $conf_1$. L'outil *GraphStream* nous permet, pour plus de visibilité à l'affichage, de repositionner interactivement les nœuds comme on le souhaite.

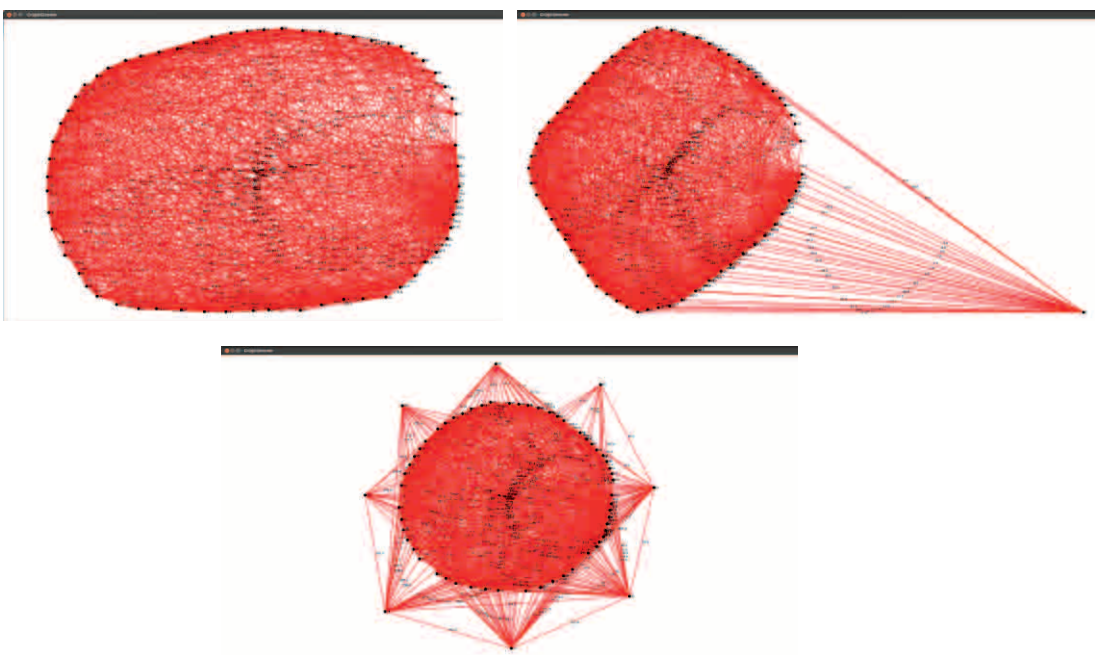


Fig-III-C2-19

Voici le mode opératoire de cet outil :

1. **calculAdjacencePart ADOT20-D6-P2-R2 2 2**
2. éditer «ADOT20-D6-P2-R2.txtTRdgs», copier-coller la *k-partition* choisie et la mettre (nœuds et vecteurs) dans «GS-MP.dgs»;
3. **java sGS** // génère le graphe d'adjacence des MM_{k-p} entre eux pour la *k-partition* retenue;
4. **cp ADOT20-D6-P2-R2.txtTRPdgs GS-MP.dgs**
5. **java sGS** // génère le graphe d'adjacence de toutes les *k-partitions* entre elles.

Sur la figure *Fig-III-C2-20*, nous avons l'organisation des fichiers (entrée/sortie) :

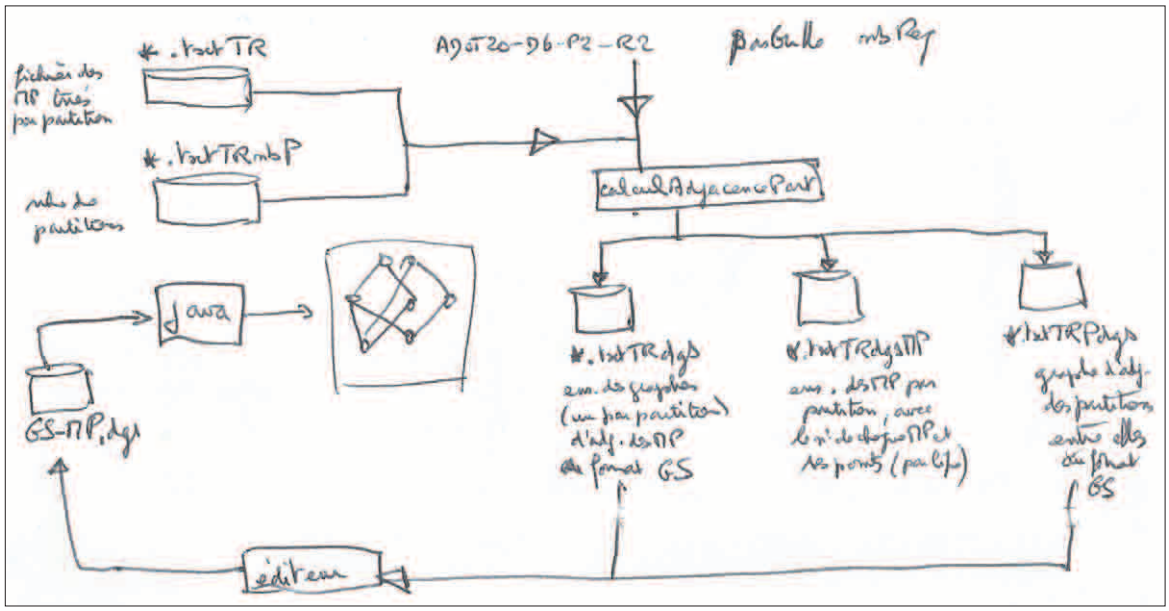


Fig-III-C2-20

ADJACENCE ENTRE DEUX *K-PARTITIONS* $CONF_I$ ET $CONF_J$

Le troisième programme «calculAdjacencePart2a2.c» calcule les adjacences entre deux *k-partitions* choisies. Cela permet de comprendre les différents passages (distance de 1) entre *k-partitions* adjacentes.

Pour obtenir le graphe d'adjacence entre les partitions $conf_i$ et $conf_j$, on lance ce programme qui génère, sur notre exemple, le fichier «ADOT20-D6-P2-R2-Parti-j.dgs».

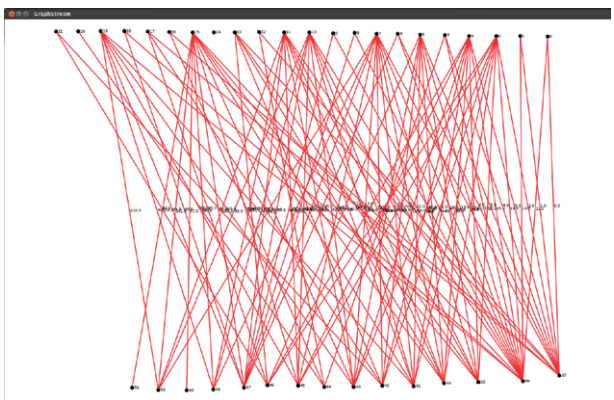


Fig-III-C2-21

Ensuite il faut renommer ce fichier en «GS-MP.dgs» et lancer la machine virtuelle *java* (**java sGS**) pour l'affichage du graphe.

Sur la figure *Fig-III-C2-21* nous avons l'adjacence de la partition optimale $conf^*_1$ avec la configuration suivante $conf_1$. Les MM_{2-p} de $conf^*_1$ sont en haut et ceux de $conf_1$ en bas.

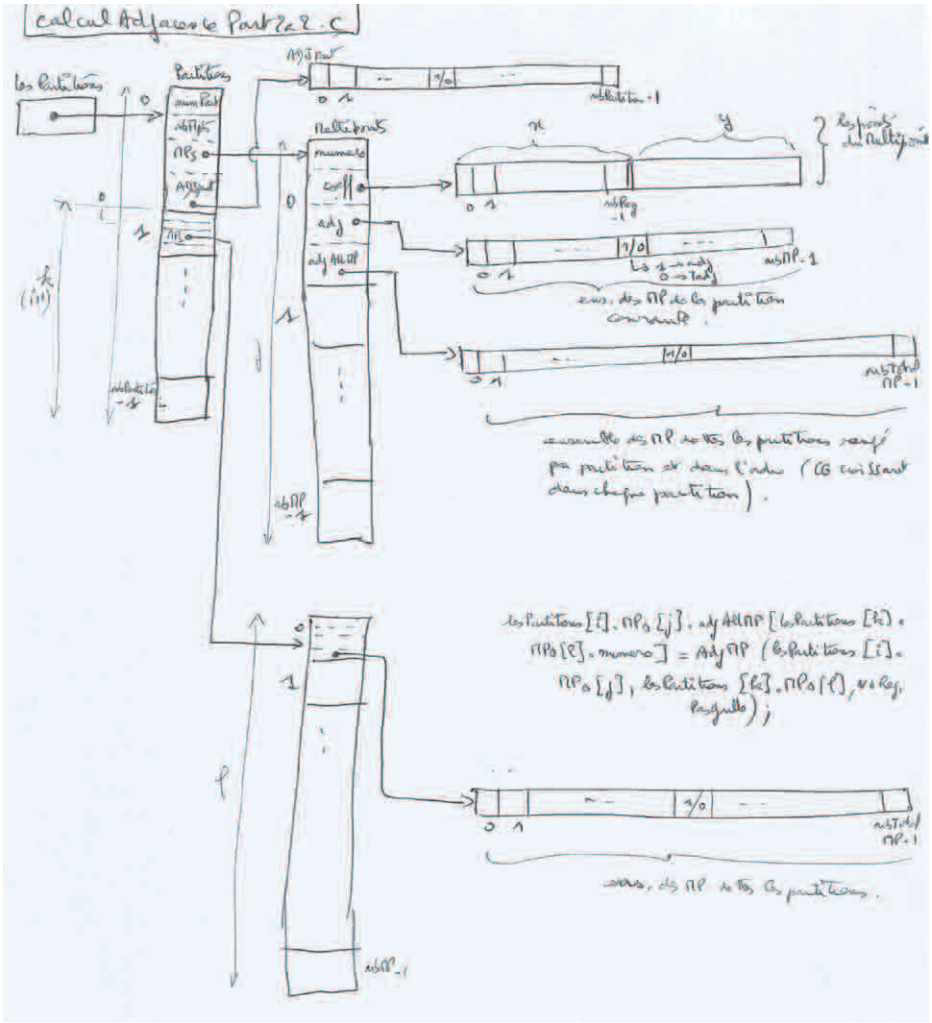


Fig-III-C2-22

Les structures de données internes (cf. la figure *Fig-III-C2-22*) au programme utilisent largement l'allocation dynamique de mémoire (*malloc*).

En résumé de cette partie voici l'ensemble des calculs qui peuvent être lancés :

```
// on lance le calcul des MM_k-p
prompt> calculCGstar-v5.sh
     ADOT20 D6 P2 R2
     points «utiles» (o/n) :
     CG*k-1 :

// on calcule le nombre de k-partitions et les MM_k-p associés
prompt> GenVector-v2.sh ADOT20-D6 P2 3

// on génère le graphe d'adjacence des MM_k-p par k-partition
prompt> CalculAdjacenceMP ADOT20-D6-P2-R3 pasGrille nbReg

// ou avec en plus le graphe d'adjacence des partitions entre elles
prompt> calculAdjcencePart ADOT20-D6-P2-R3 pasGrille nbReg

// ou avec les graphes d'adjacence des k-partitions deux à deux sur leur MM_k-p
prompt> calculAdjacencePart2a2 ADOT20-D6-P2-R3 pasGrille nbReg
```

III-C3) Un générateur de figures primitives

Pour visualiser les multi-points MM_{k-p} «utiles» nous avons construit des petits scripts d'affichage dans *gnuplot*. Nous partons des fichier «.txtTRV» calculés par «GenVector-v2.sh». Ils comportent les MM_{k-p} rangés par *k-partition* «utile» $conf_l$ en partant de la configuration optimale $conf^*_1$ (pour 'k' donné) jusqu'à la dernière.
Si l'on prend l'exemple de la lettre 'A' dans la police ADOT, avec vingt points-motif, sur une grille de trente-six points (D=6) et un *pas* de 2, nous obtenons les résultats suivants :

k	2	3	4	5	6	7	8	9	10
nb $conf_l$ utiles	64	488	183	450	492	663	230	149	133

Pour visualiser chacune de ces $conf_l$ «utiles» il suffit de lancer dans *gnuplot* :
gnuplot> **plot "ADOT20-D6-P2-R2.txtTRV" index 0 u 1:2:3:4 with vectors nohead**
pour la $conf^*_1$ (cf. la figure *Fig-III-C3-23*), puis «index 1» pour la 2e *k-partition* (cf. la figure *Fig-III-C3-24*), «index 2» pour la 3e (cf. la figure *Fig-III-C3-25*),…

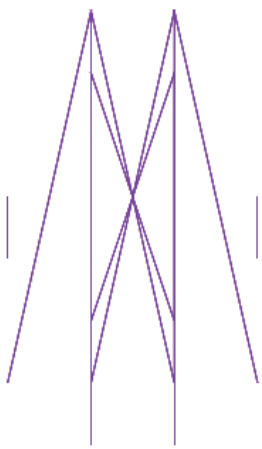


Fig-III-C3-23

Fig-III-C3-24

Fig-III-C3-25

La première démonstration consiste à visualiser l'affichage de tous les MM_{k-p} «utiles» des différentes *k-partitions* présentes dans le fichier «.txtTRV». Le script correspondant «scriptMP-R2.gp», avec k=2 (R2) est le suivant :

```
prompt> cat scriptMP-R2.gp
a=a+1
plot "ADOT20-D6-P2-R2.txtTRV" index a u 1:2:3:4 with vectors nohead
system "sleep 0.5"
if (a<63) reread
prompt>
```

Dans *gnuplot* un bloc est un ensemble de lignes de données consécutives. Deux blocs différents sont séparés par deux lignes blanches consécutives. La valeur de la variable *index* indique le numéro du bloc traité pour l'affichage. De cette façon «index 0» correspond à l'affichage de tous les MM_{k-p} «utiles» de $conf^*_1$.

Il y a autant de scripts différents que de valeur de 'k' (nombre de régions). Dans chacun d'eux le «if» s'arrête avec le nombre de MM_{k-p} «utiles» présentés dans le tableau ci-dessus.
Pour lancer la démonstration qui affiche tous les MM_{k-p} «utiles» par *k-partition* (de l'optimale vers la «moins bonne») il suffit de lancer dans *gnuplot* les instructions suivantes :

```
gnuplot> load "config-gnuplot.gnu"
gnuplot> a=0
gnuplot> load "scriptMP-R2.gp"
```

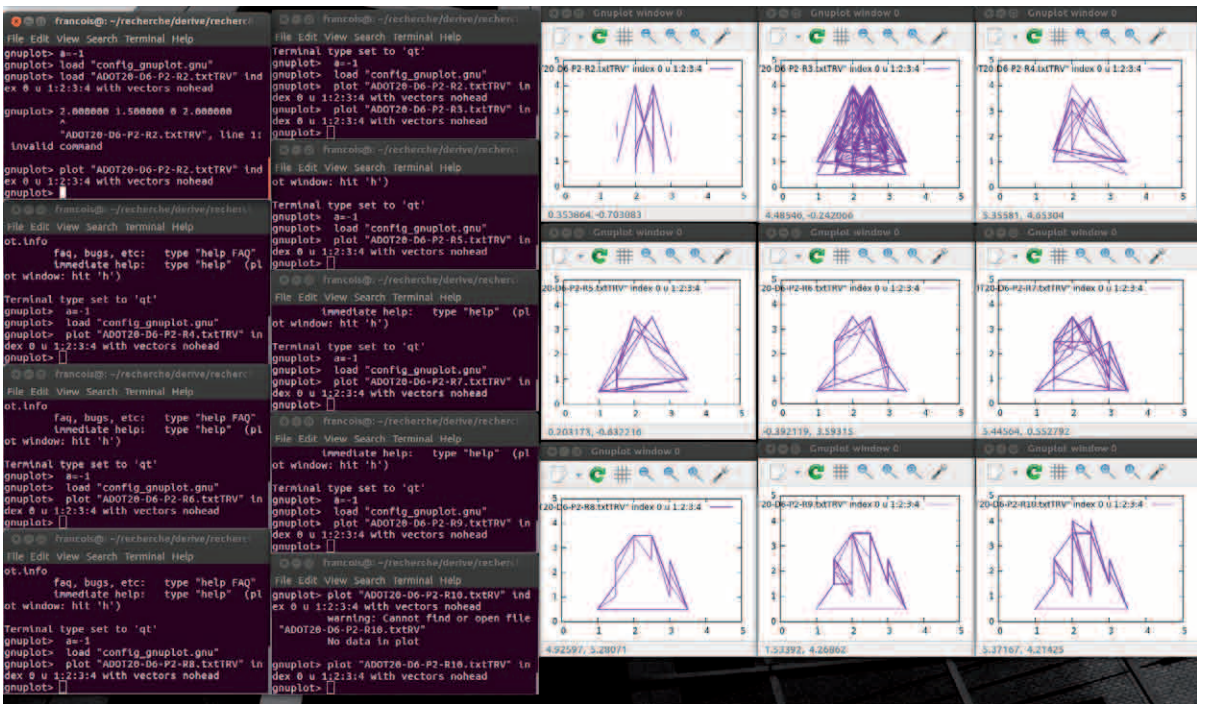


Fig-III-C3-26

Sur la figure *Fig-III-C3-26* nous avons la dernière configuration $conf_1$ affichée par le script «scriptMP-Ri.gp», avec 'i' allant de 2 à 10. Sur la figure *Fig-III-C3-27* nous avons les soixante-quatre *k-partitions* «utiles» de ADOT20-D6-P2-R2, en commençant en haut à gauche par $conf^*_1$, pour finir en bas à droite avec la dernière *k-partition* «utile».

Une autre démonstration («scriptMPmetaR2.gp») permet l'affichage en boucle des différentes *k-partitions*:

```
prompt> cat scriptMPmetaR2.gp
ab=ab+1
a=-1
load «scriptMP-R2.gp»
if (a<11) reread
prompt>
```

La valeur 11 correspond au nombre de fois que l'on va lancer «scriptMP-R2.gp». Si l'on veut faire l'affichage en parallèle avec des valeurs de 'k' différentes, il faut régler cette valeur (11) pour synchroniser les différents affichages qui comportent un nombre de configuations très variables. Voici les valeurs de synchronisation de R2 à R10 : R2(11), R3(1), R4(3), R5(1), R6(1), R7(1), R8(3), R9(3) et R10(4).

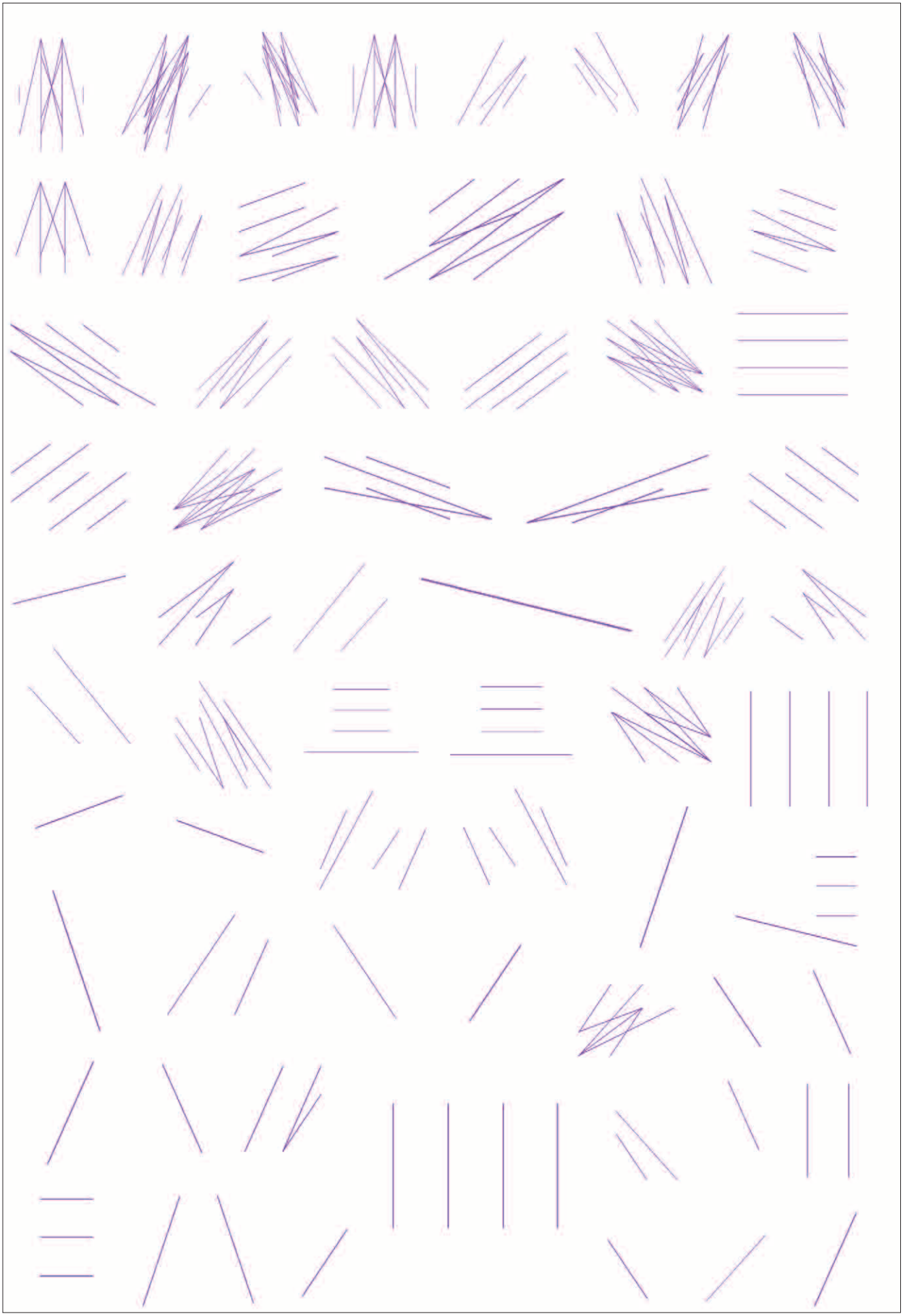


Fig-III-C3-27

Sur la figure *Fig-III-C3-28* nous avons les cinquante-six premiers MM_{3-p} correspondant à la lettre 'A' (fonte DOT) sur une grille de trente-six points (D=6). Le triangle en haut à gauche est le MM^{*}_{3-0},avec $CG(sol_p)=12,76$, alors que le dernier triangle correspond à MM_{3-55}, avec $CG(sol_p)=16,65$.

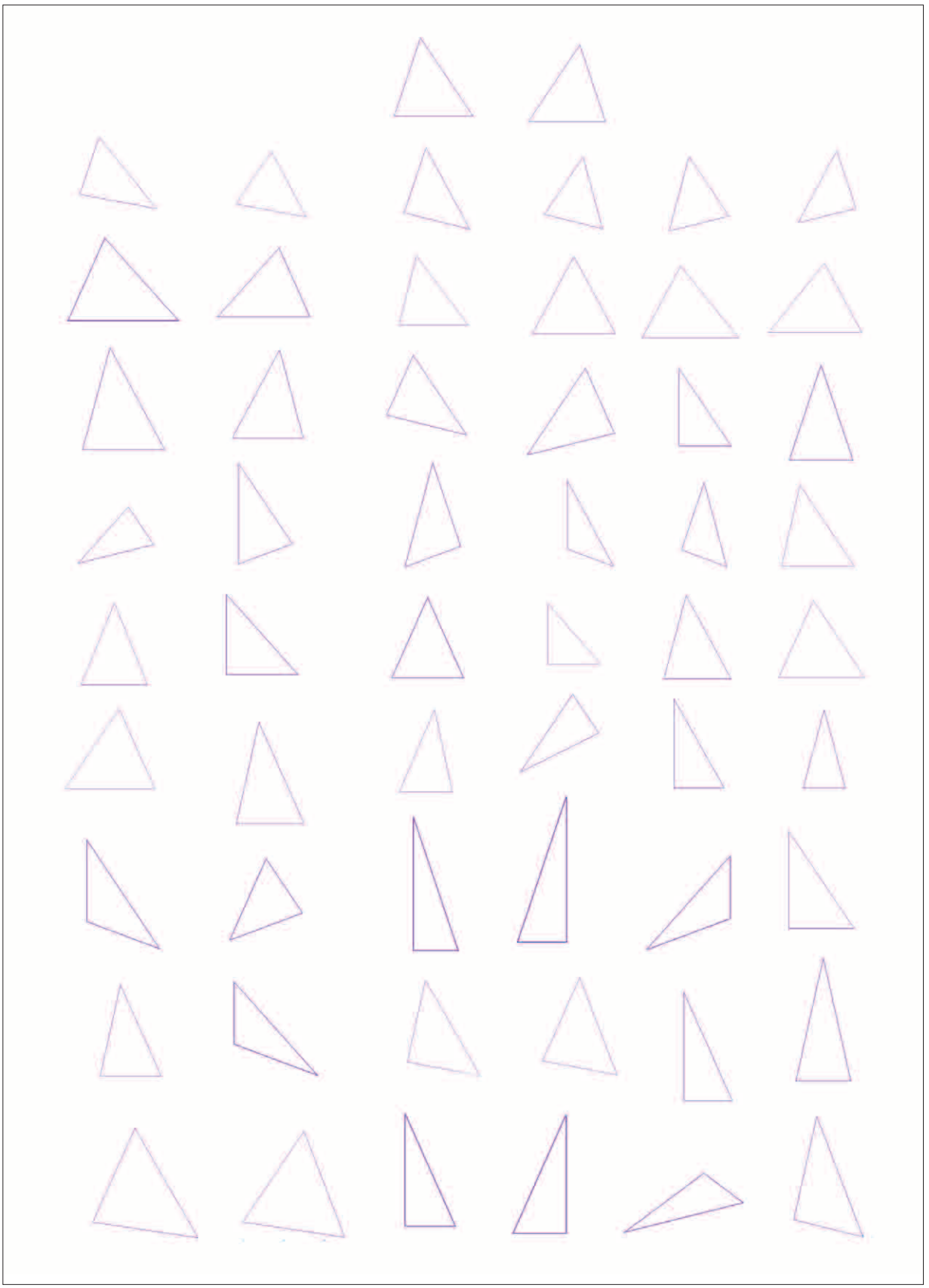


Fig-III-C3-28

III-C4) Organisation pour traiter les lettres de l'alphabet

Pour mettre en œuvre le traitement des lettres de l'alphabet avec les outils que nous venons de voir (dans la partie III-C), il est souhaitable de mettre en place une méthode de travail.

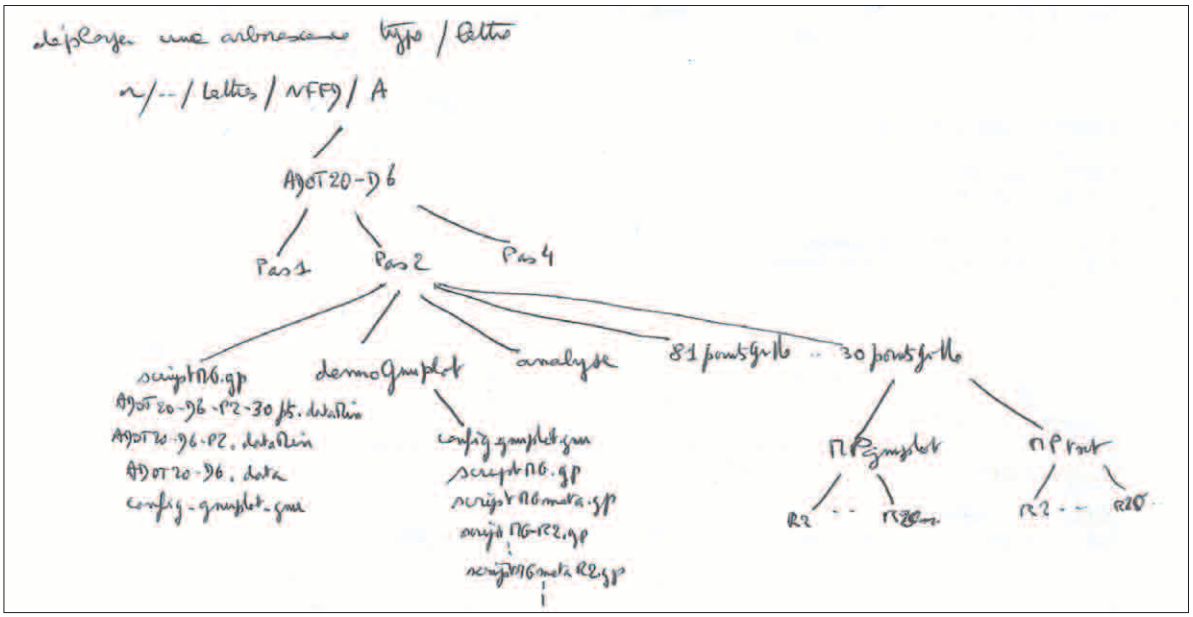


Fig-III-C3-28

Il faut commencer par choisir un *pas* de grille (1,2,4,8,…), puis on se place dans le répertoire correspondant (Pas1, Pas2,…) et on lance les programmes suivants :

```
// pour obtenir le fichier «.txt» des multi-points MM_k-p
prompt> calculCGstar-v5.sh

// pour obtenir le nombre de k-partitions «utiles» et le
// fichier «.txtTRV», pour gnuplot, des points de tous les
// MM_k-p classés par k-partition «utile» depuis conf*_1
// à la dernière
prompt> GenVector-v2.sh ADOT20-P2-R2.txt 2

// nombre de MM_k-p de conf*_1
prompt> cat ADOT20-P2-R2.txtTRS|wc -l

ensuite on recopie les fichiers «.txt*» dans le répertoire
…/Pas2/30pointsGrille/MPtxt/R2 (pour les fichiers «R2»);
on recopie aussi le fichier «.txtTRV» dans …/
Pas2/30pointsGrille/demoGnuplot; on va dans ce répertoire
et on lance gnuplot;

au préalable on aura mis à jour le script d'affichage de
tous les MM_k-p présents dans «.txtTRV»:
 → scriptMP-R2.gp (affichage de tous les MM_k-p en boucle);
 → scriptMPmetaR2.gp (affichage en boucle du script précédent).

Si on fait des photos du résultat de gnuplot, on les glissera
de Desktop vers …/Pas2/30pointsGrille/MPgnuplot/R2.
```

Étude approfondie et résultats

III-D

Nous avons vu dans la partie III-A que le nombre de Stirling de 2e espèce S(n,k) nous donne le nombre de *k-partition* pour 'n' éléments d'un ensemble, de façon «désincarnée», hors contingences, en dehors de toute forme d'«utilité». Le nombre, très vite très grand, est livré hors sol.
La nature ne peut pas se permettre l'exhaustivité de la désincarnation. Il lui faut des contraintes, un espace d'instanciation, en somme une fonction d'utilité. Son «problème» est d'exister, donc de trouver un chemin dans l'espace des possibles pour parcourir l'espace des probables. Dit autrement la nature produit des choses que les contraintes, dans lesquelles elle évolue, lui permettent. Seul ce qui est probable arrive, même si cela nécessite des processus longs et adaptatifs. La nature ne calcule rien, elle se laisse glisser vers ses solutions en perpétuelle évolution, dans le temps et dans l'espace.
Les mathématiques et le langage nous offrent un éclairage des possibles, dans le temps et dans l'espace. Associés aux processus calculatoires (via l'informatique), ils permettent de voir où nous mettons les pieds et en particulier où il ne faut pas aller (au pied du mur de la complexité calculatoire), c.-à-d. ne pas suivre des pistes que la nature elle-même n'empruntera jamais !
La simulation informatique, comme nous l'abordons par la suite lors des heuristiques résolutives, nous permet de reproduire des phénomènes/processus naturels en jouant (accélération, contraction,…) sur l'espace et le temps des faisables/probables, nous mettant ainsi à l'abri du temps et de l'espace des possibles.
L'étude approfondie ici consiste donc à instancier/ incarner S(n,k) dans le système de l'alphabet latin et de la topologie de ses lettres majuscules.
Deux perspectives s'ouvrent à nous :

1. rendre accessible/opératoire (cf. le chapitre IV) la recherche de l'optimisation dans cette incarnation (systèmes réels) ;
2. étudier (cf. le chapitre VI) les prémisses d'un processus de représentation des connaissances ouvrant les portes à l'instanciation et à la différenciation de modèles (cf. l'article introductif sur la singularité), qui sont des caractéristiques centrales dans les systèmes complexes comme les alphabets, plus généralement le langage naturel et surtout la représentation des connaissances.

III-D1) Étude des MM_{k-p} «utiles»

Il s'agit d'études avancées, réalisées grâce aux outils vus dans la partie III-C de ce chapitre III, sur les éléments présentés précédemment autour de l'alphabet latin en lettres majuscules.

Complexité de $Card(POS_k)_D$ versus S(n,k)

La formule (7) nous donne le nombre ($Card(POS_k)$) de multi-points MM_{k-p} sur la grille (DxD). Ce nombre est très vite très grand: $Card(POS_6)_{D=6}$=1 947 792, alors que $Card(POS_2)_{D=6}$=630.

La courbe (Card(POS_k), en rose sur les figures *Fig-III-D1-1/2* est symétrique autour de son point d'inflexion (extremum atteint avec $k=D^2/2$). Avec une grille de trente-six points (D=6) l'extremum de $Card(POS_k)_{D=6}$ est obtenu avec k=18 ($D^2/2$), ce qui donne :

$Card(POS_{18})_{D=6}$=9 075 133 888.

Cette courbe ne dépend pas de 'n', mais uniquement de 'k' et de 'D'. Elle a une complexité moindre que celle de S(n,k) (en vert sur les figures *Fig-III-D1-1/2*), avec un net décalage des deux courbes sur l'abscisse ('k'). Cette dernière ne dépend pas de 'D' (taille de la grille), mais uniquement de 'n' et de 'k'.

L'abscisse, sur la figure *Fig-III-D1-1*, correspond aux différentes valeurs de 'k', alors que l'ordonnée, exprimée en million, représente à la fois $Card(POS_k)_D$ et S(n,k).

On note pour S(n,k), en fonction de 'n', les extremum suivants et leur valeur associée :

S(13,k), extremum pour k=6, avec S(13,6)=9 321 312,
S(14,k), extremum pour k=6, avec S(14,6)=63 436 373,
S(15,k), extremum pour k=6, avec S(15,6)=420 693 273,
S(25,k), extremum pour k=10, avec S(25,6)=1 203 163 392 175 387 500.

À titre de comparaison il faut aller chercher S(55,k) pour trouver l'extremum pour k=18 (la même que pour $Card(POS_{18})_{D=6}$), avec la valeur suivante :

S(55,18)=72 918 880 972 237 952 530 084 715 121 408 257 344 658 255 867 447 445.

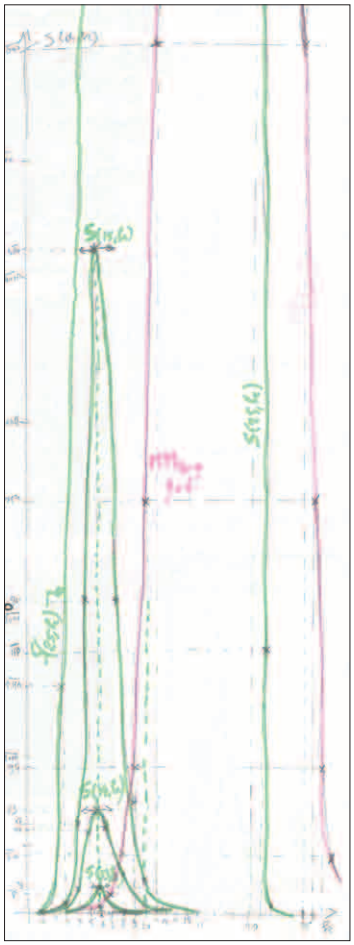


Fig-III-D1-1

Complexité de $Card(POS_k)_D$ versus 'D', à 'k' constant

Si l'on fixe 'k', par exemple à 2, l'augmentation de la taille de la grille ('D') impacte raisonnablement le nombre de *2-position* ($Card(POS_2)_D$) induit; on passe de 630 pour une grille de trente-six points à 79800 pour une grille de quatre-cents points (cf. la figure *Fig-III-D1-3*).

La valeur k=2 n'est pas choisie au hasard. Nous aurons l'occasion d'en reparler avec les algorithmes de division cellulaires proposés dans les heuristiques résolutives (cf. le chapitre IV). Nous avons déjà noté que $S(n,2)=2^{n-1}-1$ (*16*), mais aussi que $\underline{\theta}(n,2)=n(n-1)/2$ (*24*).

Le critère déterminant pour $Card(POS_k)_D$ est 'k', car sa valeur fixe la pente de $Card(POS_k)_D$. La valeur de 'D' amplifie

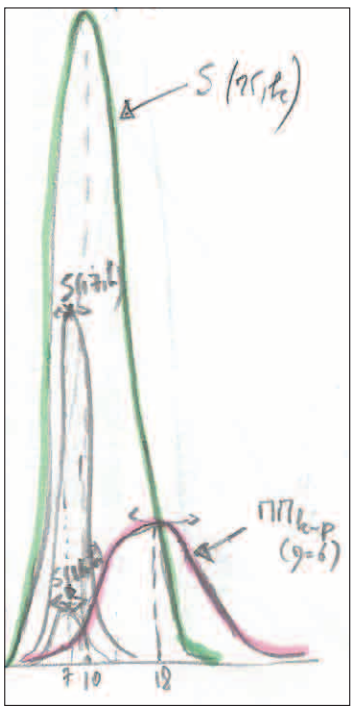


Fig-III-D1-2

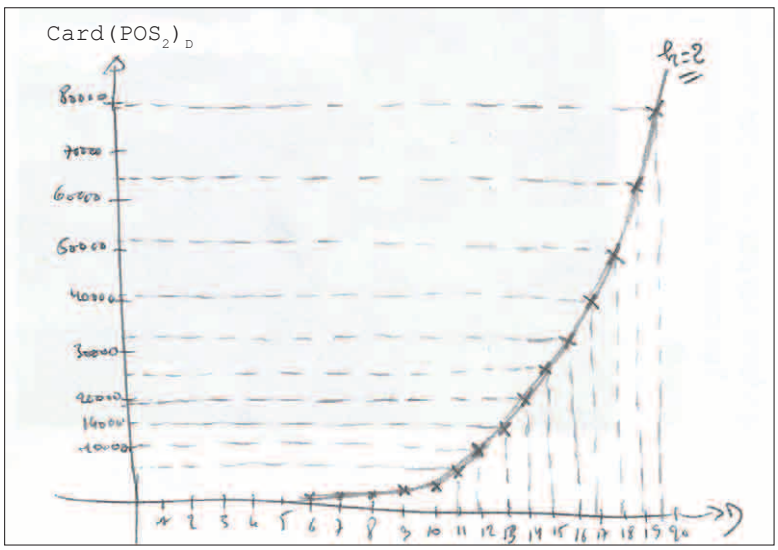


Fig-III-D1-3

cette pente. Plus la pente est forte ($k \rightarrow D^2/2$, le point d'inflexion) plus 'D' amplifie la pente et inversement.

Complexité du nombre de MM_{k-p} «utiles» versus 'k', à 'd' constant

Nous rappelons qu'un multi-points MM_{k-p} est «utile» si la solution sol_p associée à $SYS_{n,k}$ vérifie : $CG(sol_p) < CG^*_{k-1}$.
Sur le motif 'A' avec vingt-cinq points-motif et une grille de trente-six points (D=6, Pas=1) nous obtenons les valeurs suivantes :

k	1	2	3	4	5	6	7	8	9	10	11
nb MM_{k-p}	36	630	7140	58905	376992	$1,9.10^6$	$8,3.10^6$	$30,2.10^6$	$94,1.10^6$	254.10^6	600.10^6
nb MM_{k-p} «utiles»	36	208	758	661	1154	491	110	107	22	28	8
% (utiles/ total)	100	33	10	1,12	0,3	0,025	0,0013	…			

Plus 'k' augmente et plus le nombre de MM_{k-p} «utiles» diminue, pour tendre rapidement vers une quantité négligeable (cf. pour k=7 un pourcentage égal à 0.0013), alors que le nombre de MM_{k-p} (tous) explose jusqu'au point d'inflexion.

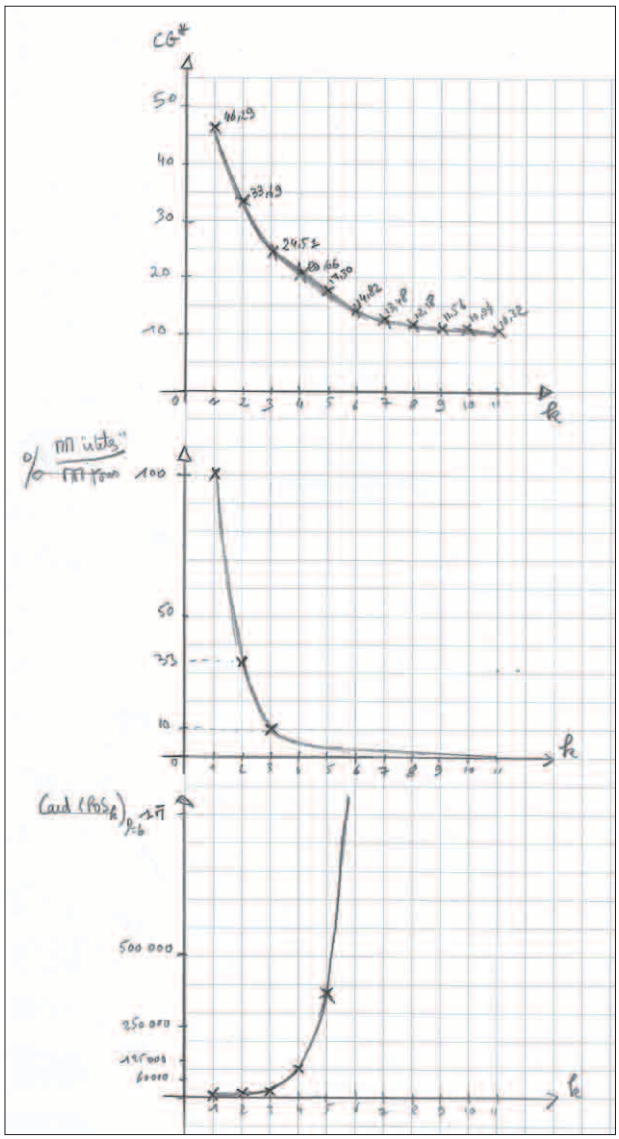


Fig-III-D1-4

Cela plaide pour ne pas trop diviser un système en régions.
Ceci est renforcé par l'évolution de $CG^*_{n,k}$ en fonction de 'k' (le nombre de régions). Comme le montre la figure *Fig-III-D1-4*, le gain en valeur absolue de $CG^*_{n,k}$ en fonction de 'k' est de plus en plus faible, pour devenir insignifiant assez rapidement (sur notre exemple avec k=11 pour une valeur pouvant aller jusqu'à 25).
Le nombre de MM_{k-p} «utiles» ou non, ne doit pas être trop faible non plus, car cela signifierait que les chances de tomber sur l'un d'entre eux seraient très faibles. Sur notre exemple (A25-D6) le nombre de MM_{k-p} «utiles» est relativement important jusqu'à k=6, au-delà il devient trop faible.
Si l'on se base sur la profondeur 'k' lors de l'analyse d'un motif, dans l'optique de mettre en place un processus de différenciation entre motifs, comme nous le verrons au chapitre VI, nous voyons se dessiner le prix à payer.

Une analyse profonde demandera de gros efforts (dans le cas d'une approche exhaustive) pour des résultats relativement modestes en terme de distanciation. D'où l'intérêt de définir des heuristiques capables de travailler directement avec les MM_{k-p} «utiles», accélérant considérablement la recherche pour un rapport qualitatif acceptable (principe d'économie de ressources).

IMPACT DE 'K' SUR L'ÉVOLUTION DE CG

La résolution de $SYS_{n,k}$ conduit, pour 'k' donné, à définir un ensemble de multi-points «utiles» MM_{k-p} et autant de solutions sol_p associées. Chacune d'elles donnant une valeur $CG(sol_p)$. On peut classer toutes ces solutions en fonction de CG (fonction d'utilité) en partant de la meilleure (CG^*) jusqu'à la moins bonne (cf. les figures *Fig-III-C1-8/9/10/11* de la partie III-C1). C'est bien entendu les meilleures solutions qui nous intéressent, mais pas uniquement, comme nous le verrons lors du processus de différenciation.

On peut regarder les courbes des *k-partitions* optimales ($conf^*_1$), pour 'n' donné et 'k' variable, afin de voir l'influence de 'k' dans la dynamique des valeurs de CG des solutions obtenues.

k	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$CG^*_{k,20}$	18,74	12,76	10,92	9,37	8,34	7,32	6,65	6,21
a=n° MM_{k-p} (au changement de pente)	30	400	100	170	300	330	100	80
b=nbre MM_{k-p} «utiles»	335	1546	898	1962	1627	2044	553	308
% (a/b)	8,96	25,87	11,14	8,66	18,44	16,14	18,08	25,97

Sur l'exemple 'A' avec D=6, vingt points-motif et Pas=2, sur la figure *Fig-III-D1-5* nous avons en abscisse les numéros d'ordre (de 1 à $Card(POS_k)_D$) des MM_{k-p} triés en fonction de la valeur $CG(sol_p)$ de la solution sol_p associée à chacun d'eux. Pour chaque courbe (de k=2 à k=9), le premier numéro correspond donc à $CG^*_{n,k}$. Par construction les courbes sont strictement croissantes.

On peut noter sur ces courbes l'évolution de CG. Dans toutes les configurations les valeurs de CG se dégradent rapidement avant d'atteindre un point à partir duquel la dégradation (augmentation de CG) est moindre en devenant quasi linéaire.
Nous avons donné en abscisse les valeurs approximatives de ces points de changement significatif de pente, ainsi que le rapport entre la coordonnée en 'x' du point correspondant et le nombre de MM_{k-p} «utiles» considérés dans la *k-partition* $conf^*_1$ courante.

Si l'on se place dans la recherche de l'optimum dans la configuration $conf^*_1$ pour 'k', ces courbes donnent une idée de la forme du gradient d'optimalité local à chaque configuration. En partant de n'importe quel MM_{k-p} de cette configuration la descente du gradient suit au pire ces courbes. En réalité la topologie du motif conduit à des proximités entre MM_{k-p} telles (cf. la notion de distance entre MM_{k-p}) que cette descente peut être accélérée (sauts dans ces courbes).

Nous retrouvons les mêmes types de courbes dans les autres configuration $conf_1$ pour 'k' donné.

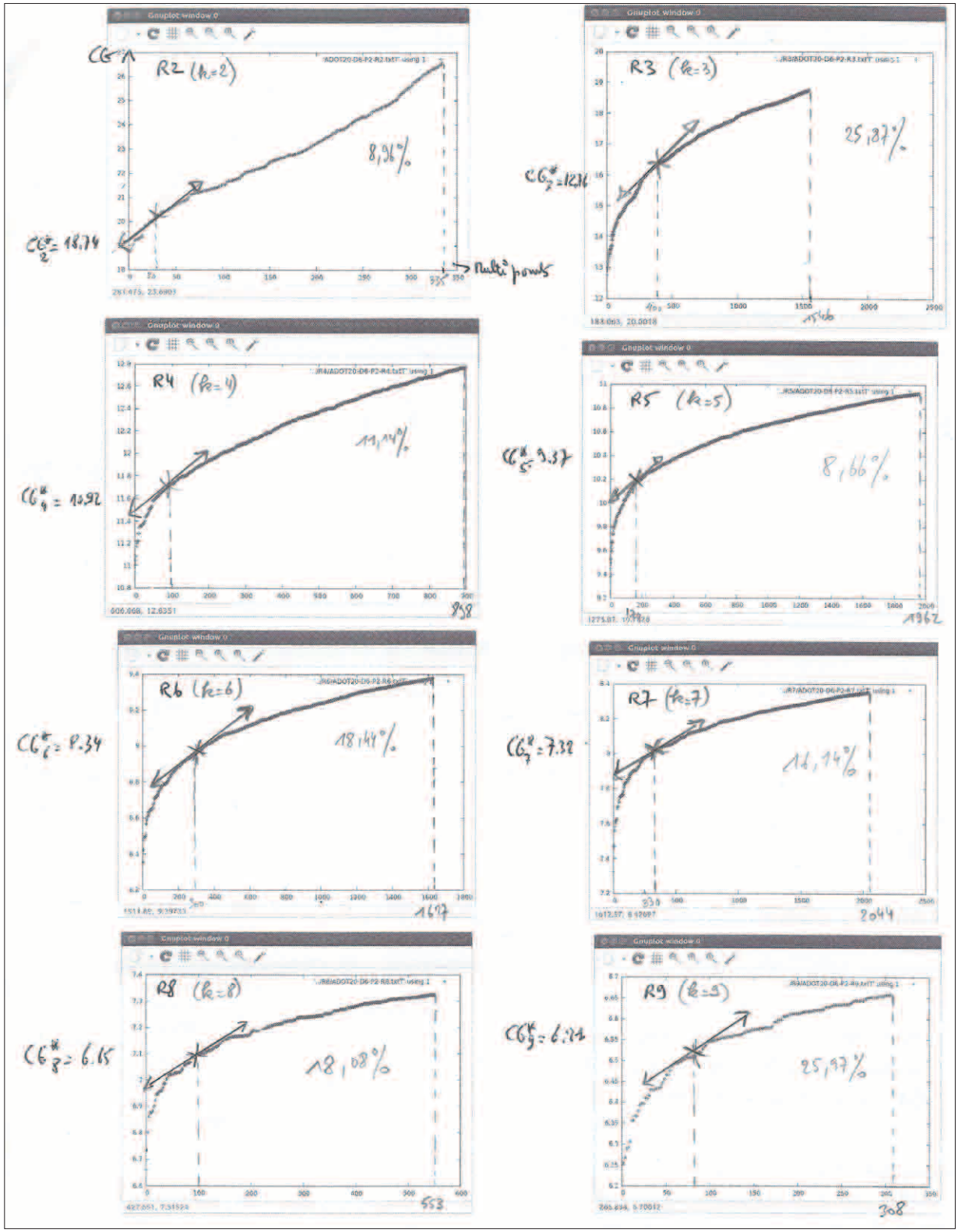


Fig-III-D1-5

Nous notons au passage que le nombre de MM_{k-p} «utiles», dans une configuration donnée, varie en fonction de 'k' et de la topologie du motif étudié. Nous savons que le nombre de MM_{k-p} augmente en fonction de 'k' jusqu'à son point d'inflexion ($k=D^2/2$) quelque soit le motif étudié. D'où l'intérêt de travailler avec les MM_{k-p} «utiles» qui, outre le fait qu'ils sont peu nombreux, n'augmentent pas directement avec 'k', mais en fonction de la topologie du motif étudié.

BASSINS D'OPTIMALITÉ LOCAUX DANS LES RÉGIONS DES *K-PARTITIONS*

Nous venons d'évoquer les gradients d'optimalité pour le parcours des MM_{k-p} appartenant à une même configuration $conf_1$. Or il faut se souvenir qu'un MM_{k-p} est constitué de 'k' points qui peuvent être modifiés (déplacement des serveurs dans C_nS_k) simultanément au cours de la recherche de l'optimum. Nous avons défini la notion de distance entre MM_{k-p}. Le problème des heuristiques résolutives est de synchroniser ou non les déplacements simultanés des points d'un MM_{k-p} donné, via les serveurs associés.

Les formules (*1*) et (*2*) (cf. la partie III-A) nous montrent que la valeur $CG(sol_p)$ se décompose en autant de CG locaux que de régions ('k') du MM_{k-p} courant.
Qu'en est-il de la dynamique des gradients d'optimalité locaux associés à ces MM_{k-p} ?

Sur la figure *Fig-III-D1-6* nous avons les huit premières *2-partitions* en partant de l'optimale $conf^*_1$ (Part-*) jusqu'à la 8e *2-partition* (Part-7).
En abscisse nous avons l'ensemble des MM_{2-p} de chaque *k-partition* et en ordonnée nous avons les valeurs de $CG(sol_p)$ décomposées suivant chacune des deux régions de chaque MM_{2-p}.
La courbe supérieure représente $CG(sol_p)$ pour chaque MM_{2-p}, alors que les deux autres courbes représentent les valeurs locales CG liées à chaque région/point du MM_{2-p}.
Conformément à ce que nous avons vu précédemment $CG(sol_p)$ suit bien un gradient d'optimalité dans chaque *k-partition* (Part-*, Part-1,…).

En revanche la figure *Fig-III-D1-6* nous montre (en rose) qu'il existe des points bas (puits locaux) dans les régions. Les serveurs ne peuvent pas suivre des gradients d'optimalité locaux indépendemment les uns des autres. L'optimal de la somme ($CG(sol_p)$) n'est pas forcément égal à la somme des optimums locaux (CG).

On peut quand même observer que les courbes des régions ont une forme de symétrie entre elles. En effet puisque la somme décroit, si l'une croît, l'autre va nécessairement décroître. Cette symétrie sera d'autant plus marquée que le gradient global est linéaire.

Chaque *k-partition* possède un bassin qui lui est optimal. Nous parlons de sous-optimalité par rapport à $SYS_{n,k}$. Seule $conf^*_1$ (Part-*) a le bassin optimal pour $SYS_{n,k}$.
On peut parcourir les MM_{k-p} jusqu'à MM^*_{k-p} ; on peut également parcourir les MM_{k-p} d'une *k-partition* donnée jusqu'à l'optimum local à cette dernière, puis passer à une autre *k-partition*, jusqu'à $conf^*_1$ et son optimal global. Dans ce dernier cas, plus le nombre de *k-partitions* est grand et plus la chance d'arriver sur $conf^*_1$ est faible. Il faudrait trouver une méta-descente de gradient qui tienne compte de l'ensemble des *k-partitions* «utiles» et qui joue sur les discontinuités (impossibilité de passer directement d'une *k-partition* à une autre) dans les *k-partitions*.

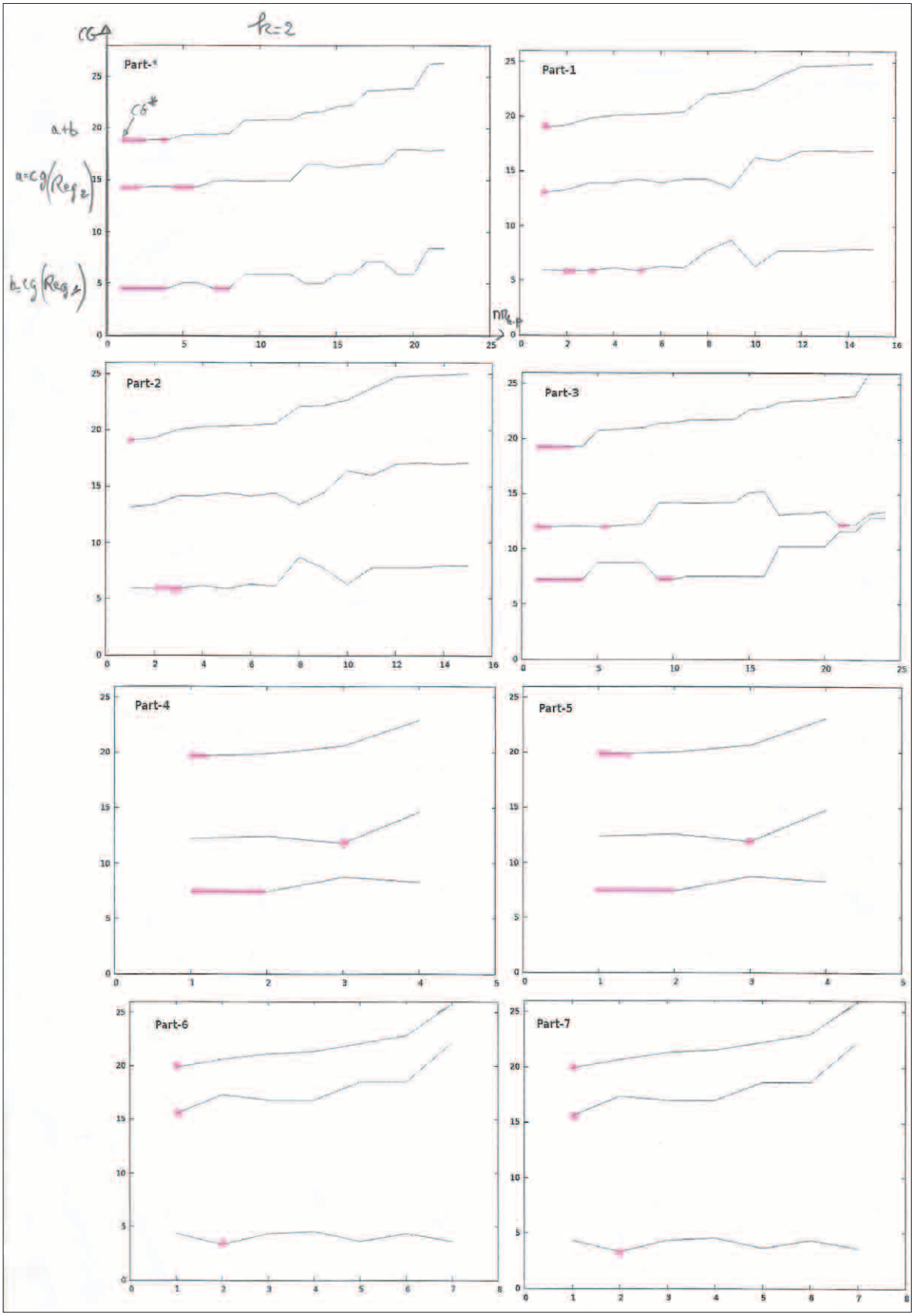


Fig-III-D1-6

RÔLE DU *PAS* DE LA GRILLE SUR LES CALCULS DE $CG_{N,K}$

Nous avons regardé les résultats obtenus pour le motif 'A' avec vingt points-motif, sur une grille de dimension D=6 en prenant respectivement un *Pas* de 1, 2 et 4.

Nous obtenons respectivement une grille de trente-six points, de cent-vingt-et-un points et de quatre-cent-quarante-et-un points.
Plus le *pas* est grand, plus la grille est fine et plus on se rapproche d'une grille continue et donc de la «vraie» valeur de $CG^*_{n,k}$. Les heuristiques résolutives (cf. la partie IV) travaillent sur une grille continue (en réel). Elles devront faire mieux que les calculs exhaustifs qui travaillent sur une grille discrétisée à cause de la complexité en temps de calcul induit.

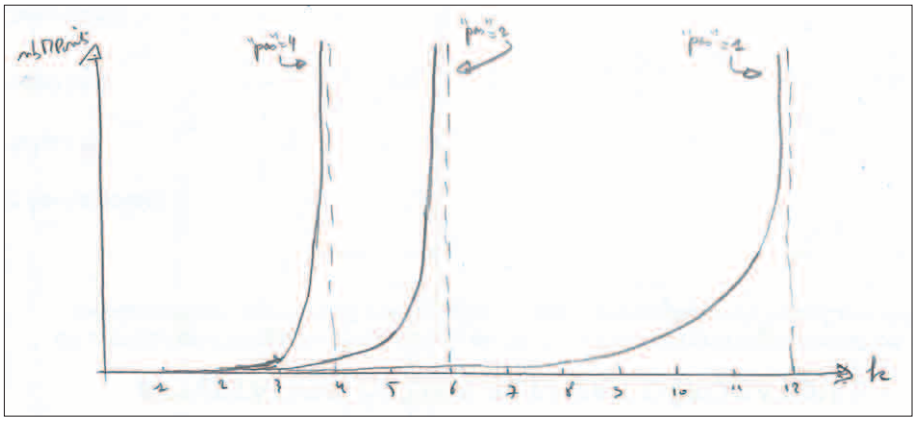


Fig-III-D1-7

On voit sur la figure *Fig-III-D1-7* l'impact du *pas* sur le nombre de MM_{k-p} (ordonnée) en fonction (abscisse) de 'k' :

$Card(Pos_{k=4})_{Pas=4}$=1 554 599 970,
$Card(Pos_{k=6})_{Pas=2}$=3 843 323 484,
$Card(Pos_{k=12})_{Pas=1}$=1 251 677 700.

De façon assez contre-intuitive les chiffres nous montrent que la complexité des calculs pour des résolutions fines (*pas*$\geqslant$4) ne permettent pas d'atteindre des valeurs de CG^* aussi précises qu'avec des résolutions moins fines. Dans l'absolu, plus le *pas* est grand et plus la valeur calculée est précise, si elle peut être calculée. Pour filer la métaphore d'un microscope, un *pas* de faible valeur permet d'atteindre rapidement des valeurs approchées de CG^*. D'autres mécanismes, comme la prise en compte uniquement des points «utiles» de la grille, prendront le relais pour affiner les résultats obtenus.

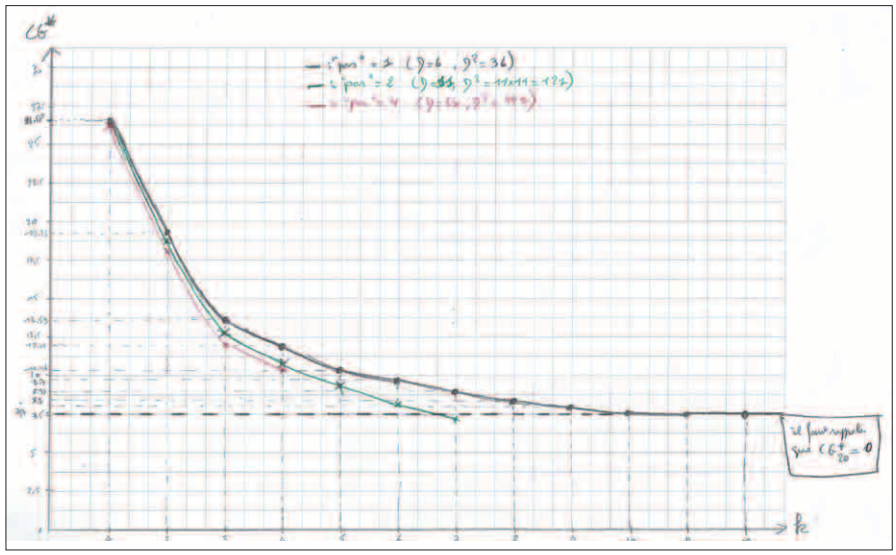


Fig-III-D1-8

Sur la figure *Fig-III-D1-8* nous obtenons les résultats suivants :

- pas=1 (en noir) : $CG^*_{20,10}=CG^*_{20,11}=CG^*_{20,12}=...CG^*_{20,20}$=7.7070, avec $Card(Pos_{k=10})_{Pas=1}$=254 186 856;
- pas=2 (en vert) : $CG^*_{20,5}$=9.3794, avec $Card(Pos_{k=5})_{D=6}$=198 792 594,
  $CG^*_{20,6}$=8.3495, avec $Card(Pos_{k=6})_{D=6}$=3 843 323 484,
  $CG^*_{20,7}$=7.3262, avec $Card(Pos_{k=7})_{D=5}$=3 477216 600;
- pas=4 (en rouge) : $CG^*_{20,4}$=10.5159, avec $Card(Pos_{k=4})_{Pas=4}$=1 354 599 970.

Malgré la parallèlisation des calculs nous n'avons pas pu atteindre $CG^*_{20,7}$ avec un pas=2 (nous avons continué avec D=5), ni $CG^*_{20,5}$ avec un pas=4 (continuer la courbe rouge de la figure *Fig-III-D1-8*).

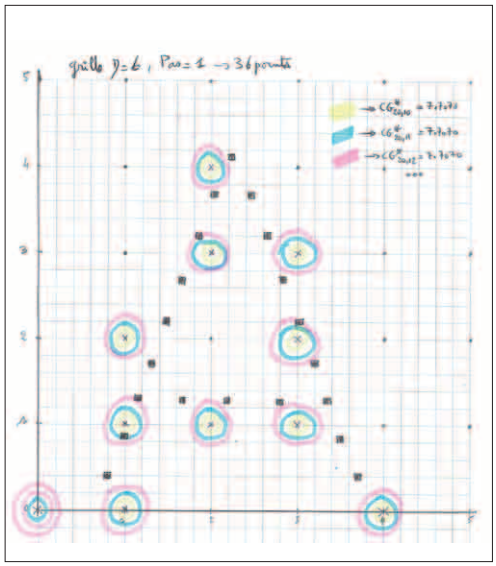


Fig-III-D1-9

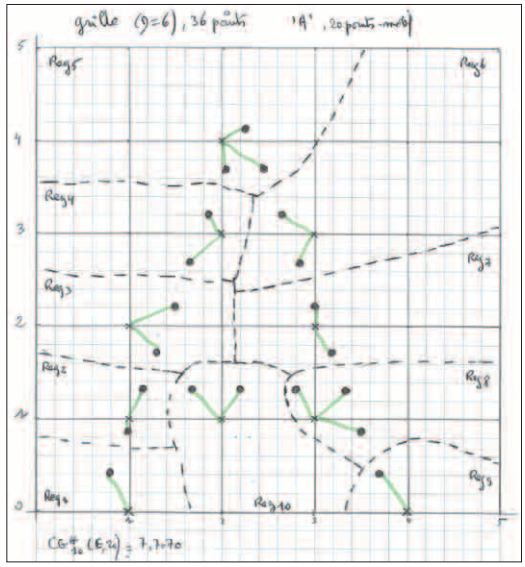


Fig-III-D1-10

Cela nécessite de parcourir 135 872 037 378 MM_{5-p}.
Pour pas=1 on atteint la meilleure valeur approchée de CG^* (7.7070) avec k=10 (cf. la figure *Fig-III-D1-9*), où l'on voit les MM^*_{k-p} (en jaune pour k=10, en bleu pour k=11 et en rose pour k=12).

On comprend mieux avec la figure *Fig-III-D1-10* le rôle de la grille dans les résultats obtenus. On voit les dix régions (k=10) avec, pour chacune d'elles, la répartition (traits vert) des points-motif de 'A'. L'alignement sur la grille des points de MM_{10-p} montre clairement les marges d'amélioration sur une grille plus fine, voire en continu. Pour faire mieux il faut passer à pas=2 et k=7 et un temps de calcul terriblement allongé. Dans nos conditions de calcul, avec un pas=4, nous n'arrivons pas à dépasser k=4 avec une valeur de CG^* nettement moins bonne. Avec de meilleures conditions de calcul nous ne ferions que reporter le problème un peu plus loin. C'est toute la problématique de la complexité algorithmique.

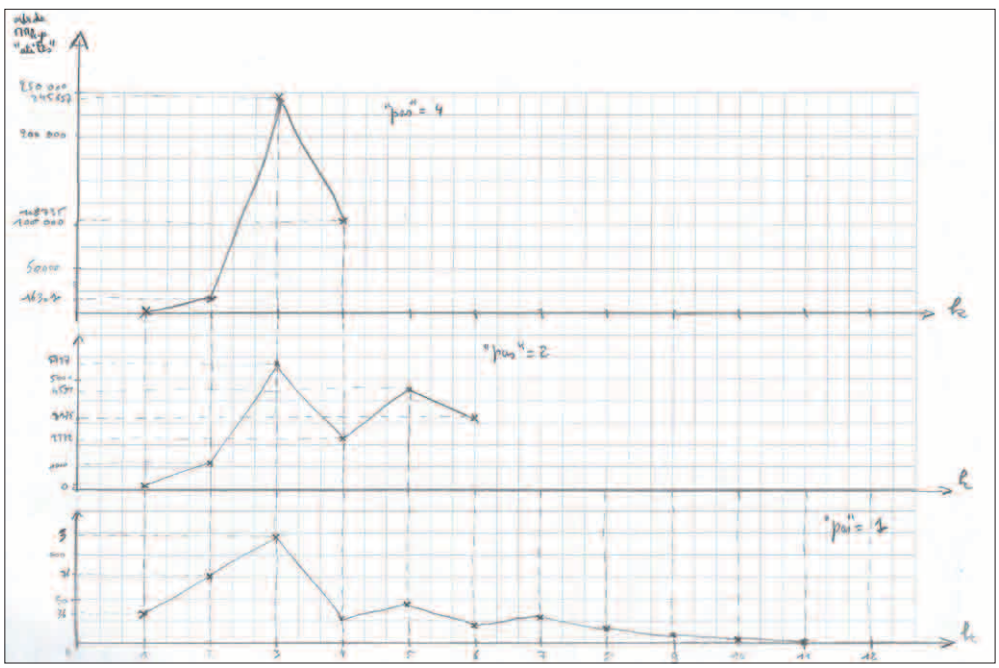


Fig-III-D1-11

Nous pouvons aussi regarder l'impact conjoint de 'k' et du *pas* de la grille sur le nombre de MM_{k-p} «utiles». Les points manquant (cf. la figure *Fig-III-D1-11*) n'ont pas pu être calculés sur notre machine. On peut néanmoins remarquer que les trois courbes ont sensiblement la même forme. Il faudrait approfondir cette étude, en particulier avec les autres

lettres de l'alphabet, pour voir que, probablement, elles correspondent à une signature du motif auquel elles se rapportent.

Rôle des points «utiles» de la grille

Dans les parties précédentes nous avons vu que S(n,k) peut être très nettement minimisée en ne considérant que les *k-partitions* $conf_l$ «utiles», elles-mêmes s'appuyant sur les multi-points MM_{k-p} «utiles», qui minimisent très fortement le nombre $Card(POS_k)$.
Un autre facteur décisif, abordé précédemment, pour diminuer le nombre des calculs est de ne considérer que les points «utiles» de la grille, c.-à-d. ceux compris dans l'enveloppe convexe du motif étudié. Cela permet d'éviter des calculs inutiles car nous savons que les MM_{k-p} qui produiront les meilleures solutions sont dans l'enveloppe convexe du motif.
Là encore il faut rester prudent car l'objectif est d'atteindre le plus rapidement possible les solutions

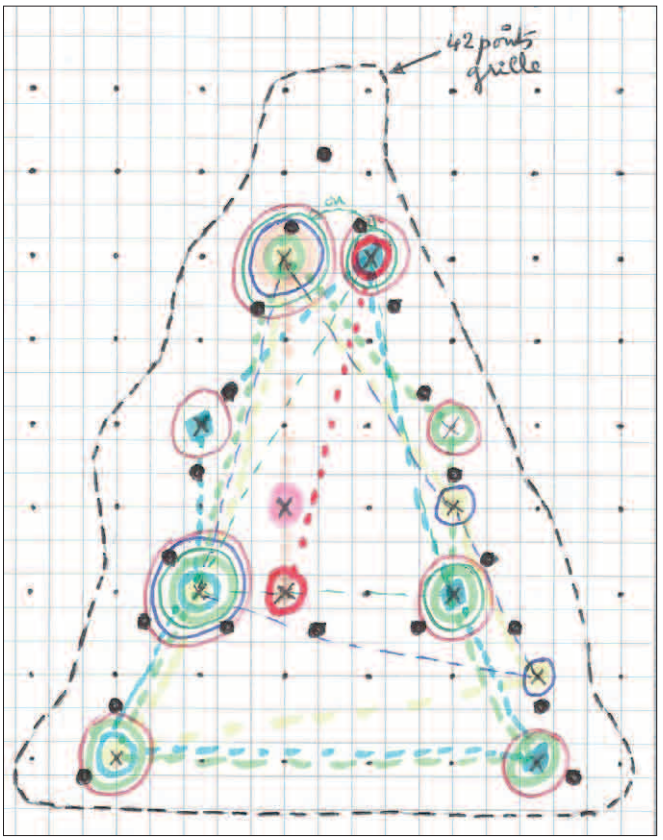


Fig-III-D1-12

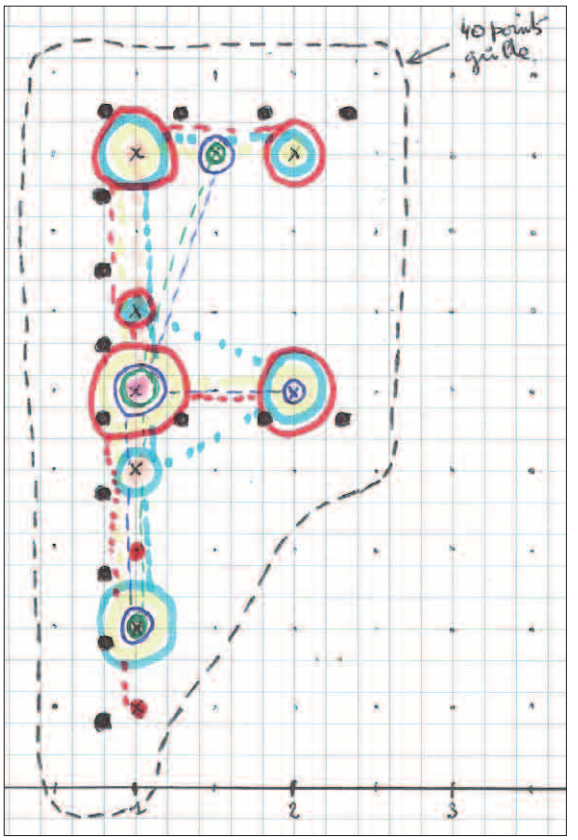


Fig-III-D1-13

optimales. Or si l'on resserre trop la couverture de recherche/déplacement autour du motif, nous pouvons obérer des chances de passage vers les MM_{k-p} recherchés. Tout et toujours affaire de compromis. Ici, comme ailleurs, le détour peut conduire plus efficacement au but recherché.
Sur les figures *Fig-III-D1-12/13* nous avons quarante-deux points-grille «utiles» pour le motif 'A' (vingt points-motif) et quarante points-grille «utiles» pour le motif 'F' (quinze points-motif) sur une grille de cent-vingt-et-un points (D=6 et pas=2).

Pour déterminer les points-grille «utiles» nous commençons par prendre D=9 avec pas=2 soit quatre-vingt-un points-grille. C'est la grille minimale qui englobe tous les points-motif. Cela nous permet de commencer les calculs.
Pour 'F' nous avons ainsi pu calculer :

$CG^*_{15,k=1}$=18.2151 (rose),
$CG^*_{15,k=2}$=11.2101 (orange),
$CG^*_{15,k=3}$=8.7062 (vert),
$CG^*_{15,k=4}$=7.2722 (bleu marine),
$CG^*_{15,k=5}$=5.9493 (jaune).

Cela nous a permis d'obtenir les MM^*_{k-p} visibles sur la figure *Fig-III-D1-13*. À partir de ces premiers résultats nous pouvons affiner l'enveloppe convexe des MM_{k-p} en ne gardant que quarante points-grille autour du motif ('F').
Nous obtenons ainsi :

$CG^*_{15,k=6}$=5.2355 (bleu ciel),
$CG^*_{15,k=7}$=4.7929 (rouge),
$CG^*_{15,k=8}$=4.5967.

Nous continuons sur ce motif 'F' avec vingt-et-un points-grille :

$CG^*_{15,k=9}$=4.4583,
$CG^*_{15,k=10}$=4.4097,
$CG^*_{15,k=11}$=4.3630,
$CG^*_{15,k=12}$=4.3198.

Le cas de k=12 est un peu particulier pour 'F' puisqu'il n'y a qu'une seule *k-partition* possible et un seul MM_{12-1} «utile», le multi-points optimal MM^*_{12}.

Sur la figure *Fig-III-D1-12* nous avons utilisé la même approche pour le motif 'A' (Pas=2) avec quatre-vingt-onze points-grille, puis quarante-deux, puis trente :

$CG^*_{20,k=1}$=28.5825 (rose),
$CG^*_{20,k=2}$=18.7408 (orange et rouge feutre),
$CG^*_{20,k=3}$=12.7671 (vert bic),
$CG^*_{20,k=4}$=10.9295 (bleu marine),
$CG^*_{20,k=5}$=9.3794 (jaune),
$CG^*_{20,k=6}$=8.3495 (bleu ciel et vert feutre),
$CG^*_{20,k=7}$=7.3262,
$CG^*_{20,k=8}$=6.6595 (rouge bic),
$CG^*_{20,k=9}$=6.2143,
$CG^*_{20,k=10}$=5.8094,
$CG^*_{20,k=11}$=5.5417,
$CG^*_{20,k=12}$=5.274,
$CG^*_{20,k=13}$=5.174,
$CG^*_{20,k=14}$=5.074,
$CG^*_{20,k=15}$=4.9886,
$CG^*_{20,k=16}$=$CG^*_{20,k=17}$=$CG^*_{20,k=18}$=$CG^*_{20,k=19}$=$CG^*_{20,k=20}$=4.9032.

Nous avons réussi à calculer 'A' sur quarante-deux points-grille jusqu'à k=10 inclus, puis sur trente points-grille jusqu'à k=20. Nous avons pu vérifier la pertinence de la réduction de l'enveloppe convexe de quarante-deux points-grille à trente, autour du motif, en calculant, après coup, toutes les valeurs de CG^*_{k-20} avec trente points-grille. Nous obtenons bien les mêmes résultats qu'avec quarante-deux points-grille.
Cela confirme le fait que les MM^*_{k-p} sont bien compris dans l'enveloppe convexe des points-motif.

Cette technique d'ajustement progressif, adaptée à chaque motif, permet d'affiner les calculs en augmentant la résolution (maille de la grille) sans trop pénaliser les temps de calculs.

De cette technique nous en tirons une propriété observée sur A20-D6-P2 (motif 'A', vingt points-motif *maille* de ½ - pas=2). Pour cela nous avons extrait dans les tableaux suivants les pourcentages du nombre de MM_{k-p} «utile» par rapport au nombre de MM_{k-p} théorique en fonction du nombre de points-grille considéré.

nb points-grille, avec k=2	**121**	**42**	**30**
nb MM_{2-p}	7260	861	435
nb MM_{2-p}/nb MM_{2-p} (121 pts)	100	11.86	5.99
nb MM_{2-p} «utiles»	1128	565	335
nb MM_{2-p} «utiles»/ nb MM_{2-p}	15.54	65.62	77

nb points-grille, avec k=3	**121**	**42**	**30**
nb MM_{2-p}	287980	11480	4060
nb MM_{3-p}/nb MM_{3-p} (121 pts)	100	3.99	1.41
nb MM_{3-p} «utiles»	5727	3138	1546
nb MM_{3-p} «utiles»/ nb MM_{3-p}	1.99	27.33	38.08

nb points-grille, avec k=4	**121**	**42**	**30**
nb MM_{4-p}	8495410	111930	27405
nb MM_{4-p}/nb MM_{4-p} (121 pts)	100	1.32	0.32
nb MM_{4-p} «utiles»	2272	1865	898
nb MM_{4-p} «utiles»/ nb MM_{4-p}	0.027	1.66	3.28

De façon évidente plus on réduit la surface de la grille, plus on réduit le nombre de MM_{k-p} ($Card(POS_k)$). Il en est de même pour les MM_{k-p} «utiles». Pour ces derniers la variation à la baisse fonctionne en sens inverse par rapport aux MM_{k-p} «ordinaires». En effet nous voyons dans les trois tableaux ci-dessus, que plus on réduit la grille et plus la quantité de MM_{k-p} «utiles» augmente. Bien que ce phénomène ait tendance à s'aplatir avec 'k' croissant, il n'en reste pas moins réel.

Propriété: On peut interpréter ces résultats en disant que plus la grille se réduit autour du motif, plus on conserve de MM_{k-p} «utiles» parmi les MM_{k-p} «ordinaires». Autrement dit, la réduction de la grille n'est pas trop rédibitoire pour l'exploration, puisqu'elle perd de moins en moins de MM_{k-p} «utiles».

Nous avons également calculé le nombre exact de *k-partitions* «utiles» ($\underline{\theta}(n,k)$) en fonction de 'k' et du nombre de points-grille retenus.

nb points-grille, avec k=2 'A', n=20, pas=2	**121**	**42**	**30**
S(n,k) nb *k-partitions*	524 287	524 287	524 287
$\underline{\theta}$(n,k) nb *k-partitions* «utiles»	191	77	64
$\underline{\theta}$(n,k)/$\underline{\theta}$(n,k) (121 pts)	100	40.31	33.51

nb points-grille, avec k=3 'A', n=20, pas=2	**121**	**42**	**30**
S(n,k) nb *k-partitions*	580 606 446	580 606 446	580 606 446
$\underline{\theta}$(n,k) nb *k-partitions* «utiles»	491	416	325
$\underline{\theta}$(n,k)/$\underline{\theta}$(n,k) (121 pts)	100	84.72	66.19

nb points-grille, avec k=4 'A', n=20, pas=2	**121**	**42**	**30**
S(n,k) nb *k-partitions*	45 232 115 901	45 232 115 901	45 232 115 901
$\underline{\theta}$(n,k) nb *k-partitions* «utiles»	252	241	183
$\underline{\theta}$(n,k)/$\underline{\theta}$(n,k) (121 pts)	100	95.63	72.62

Ces tableaux montrent, sur cet exemple, l'efficacité des *k-partitions* «utiles» ($\underline{\theta}(n,k)$) qui incarnent S(n,k) autour d'un motif donné. Nous avons vu précédemment (partie III-A3) que S(n,k) est indépendant des dimensions de la grille. Le nombre de *k-partitions* «utiles» devrait l'être tout autant, sauf qu'il tient compte des caractéristiques topologiques du motif. En diminuant le nombre de MM_{k-p} pour se restreindre aux MM_{k-p} «utiles» on diminue mécaniquement le nombre de *k-partitions* $conf_1$ pour se restreindre à celles qui sont «utiles». D'où les variations obtenues dans ces tableaux. De façon assez intuitive le nombre de ces *k-partitions* «utiles» diminue avec le nombre de points-grille retenus.
Plus 'k' augmente et moins cette diminution est importante en pourcentage. Pour k=2 il reste 40.31% des *k-partitions* sur quarante-deux points-grille, alors qu'il en reste 84.72% pour k=3 et 95.63% pour k=4. Ce phénomène perdure avec trente points-grille, mais en se tassant.
Il faudra étudier de manière plus approfondie ces grandeurs dans le processus de différenciation entre modèles, dont nous parlerons au chapitre VI.

Pour terminer cette étude autour des MM_{k-p} «utiles» et des points-grille, nous allons afficher (*gnuplot*) les MM_{k-p} «utiles» en faisant varier 'k' et le nombre de points-grilles. Nous allons appliquer ces affichages pour les motifs 'A' (ADOT-20-D6-P2) et 'F' (FDOT15-D6-P2).

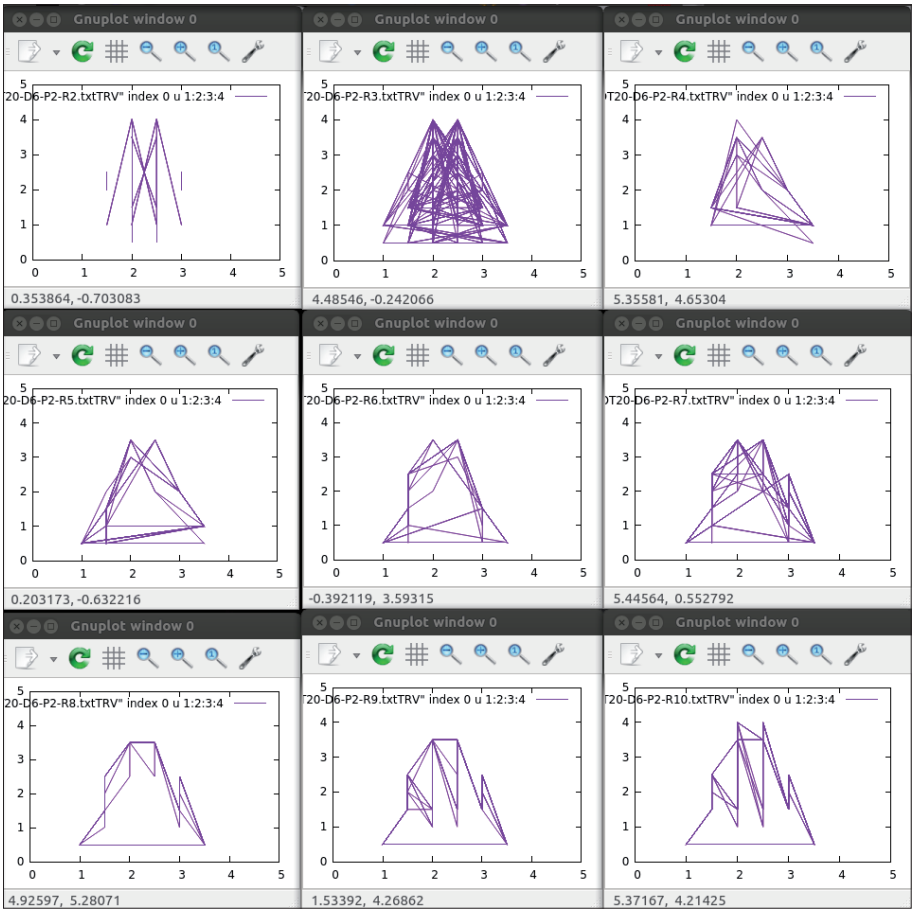


Fig-III-D1-14

MOTIF 'A'

Nous avons, sur la figure *Fig-III-D1-14*, tous les multi-points MM_{k-p} «utiles» de la *k-partition* optimale $conf^*_1$ pour les valeurs de 'k' allant de 2 (R2, en haut à gauche) à 10 (R10, en bas à droite) avec trente points-grille. Pour les MM_{k-p} avec k>2 nous avons fermé le polygone inscrit par les points de MM_{k-p}. Cet affichage nous permet d'avoir rapidement une idée du nombre de MM_{k-p} de chaque *k-partition*, ainsi qu'un apperçu de l'empreinte de ces MM_{k-p} dans le plan. C'est une sorte d'identité visuelle ou topologique.

La figure *Fig-III-D1-15* représente la même chose que la figure *Fig-III-D1-14* mais cette fois avec tous les MM_{k-p} «utiles» de toutes les *k-partitions* «utiles».
Cela confirme visuellement ce que l'on pouvait attendre avec uniquement les MM_{k-p} «utiles» des $conf^*_1$, à savoir l'apparition de l'enveloppe convexe du motif étudié.

Le troisième graphique (cf. la figure *Fig-III-D1-16*) montre avec k=3 le rôle de la grille sur la représentation visuelle des MM_{k-p} du motif. À gauche nous avons une grille de cent-vingt-et-un points, au centre une de quarante-deux points et à droite une de trente points.

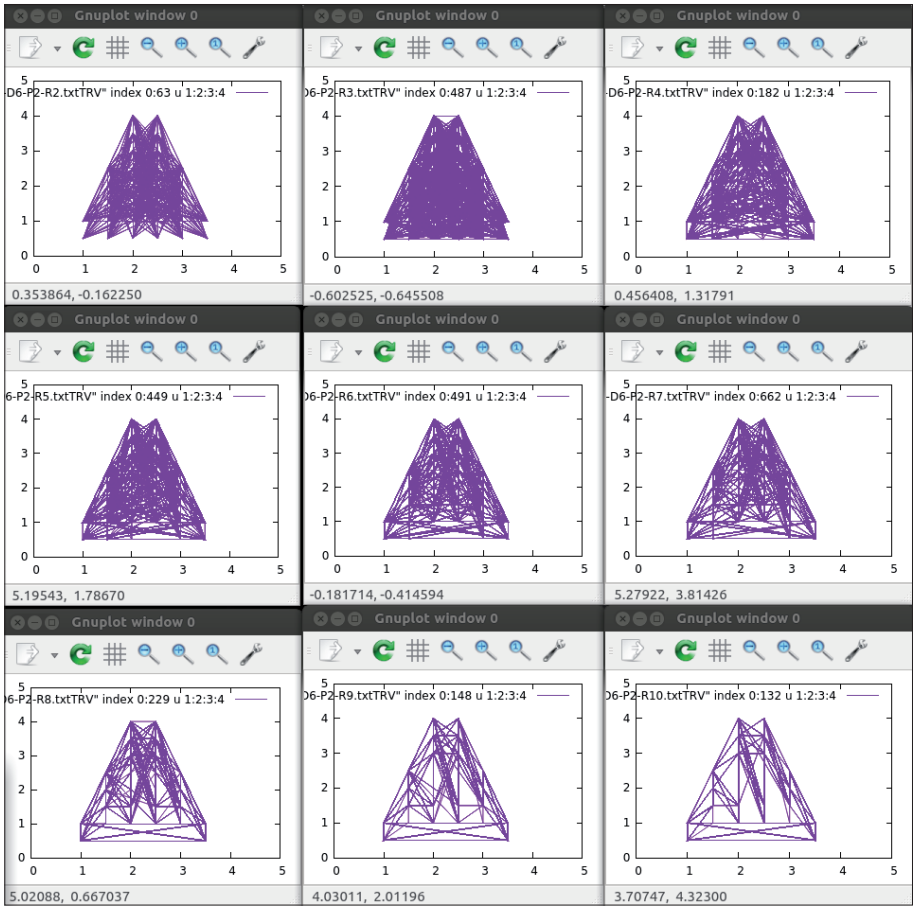


Fig-III-D1-15

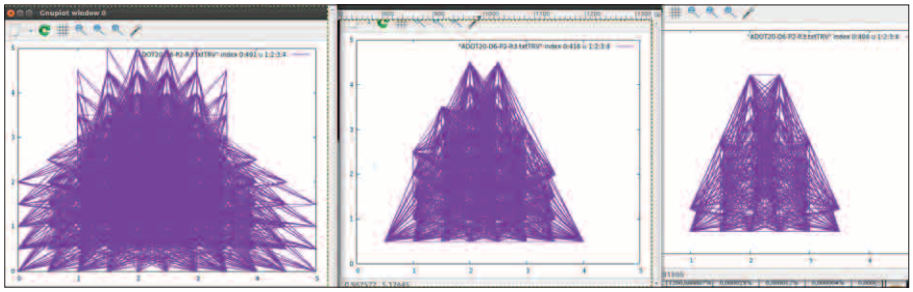


Fig-III-D1-16

Sans anticiper sur l'étude de la différenciation de modèles autour de l'alphabet latin en lettre majuscule, nous présentons les mêmes résultats avec le motif 'F'.

Motif 'F'

Nous avons, sur la figure *Fig-III-D1-17*, tous les multi-points $MM_{k\text{-}p}$ «utiles» de la *k-partition* optimale $conf^*_1$ pour les valeurs de 'k' allant de 2 (R2, en haut à gauche) à 12 (R12, en bas à droite) avec vingt-et-un points-grille. Comme pour 'A', pour les $MM_{k\text{-}p}$ avec k>2 nous avons fermé le polygone inscrit par les points de $MM_{k\text{-}p}$.
La figure *Fig-III-D1-18*, représente la même chose que la figure *Fig-III-D1-17*, mais cette fois avec tous les $MM_{k\text{-}p}$ «utiles» de toutes les *k-partitions* «utiles» toujours sur vingt-et-un points-grille.
Enfin les figures *Fig-III-D1-19/20* sont équivalentes aux figures *Fig-III-D1-17/18*, mais cette fois avec une grille

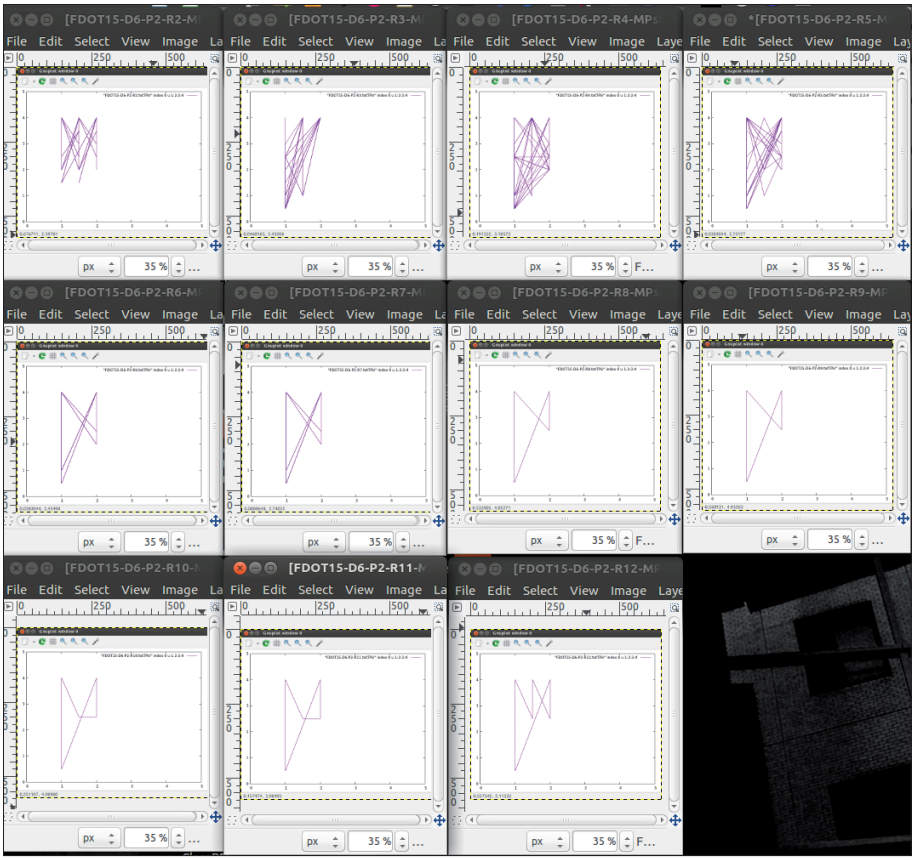


Fig-III-D1-17

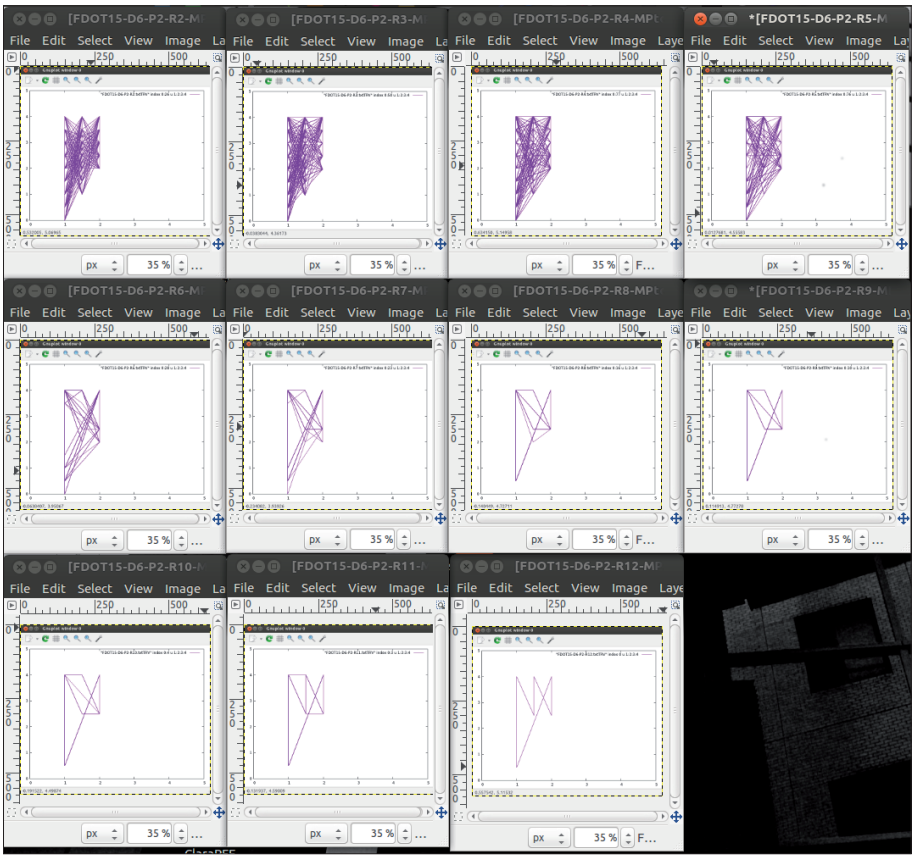


Fig-III-D1-18

de cent-vingt-et-un points-grille pour ‘k’ allant de 2 à 5, de quarante points-grille pour ‘k’ allant de 6 à 8 et de vingt-et-un points-grille pour ‘k’ allant de 9 à 12. Il est à noter que pour ces dernières valeurs nous obtenons naturellement les mêmes valeurs que sur les figures

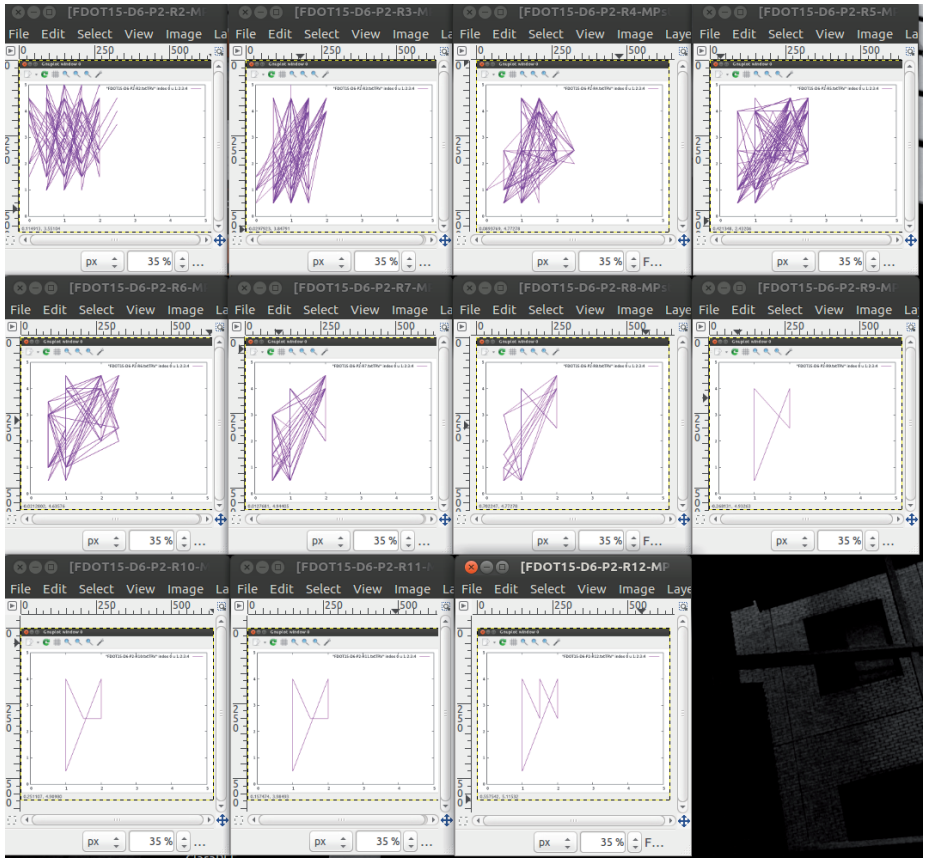


Fig-III-D1-19

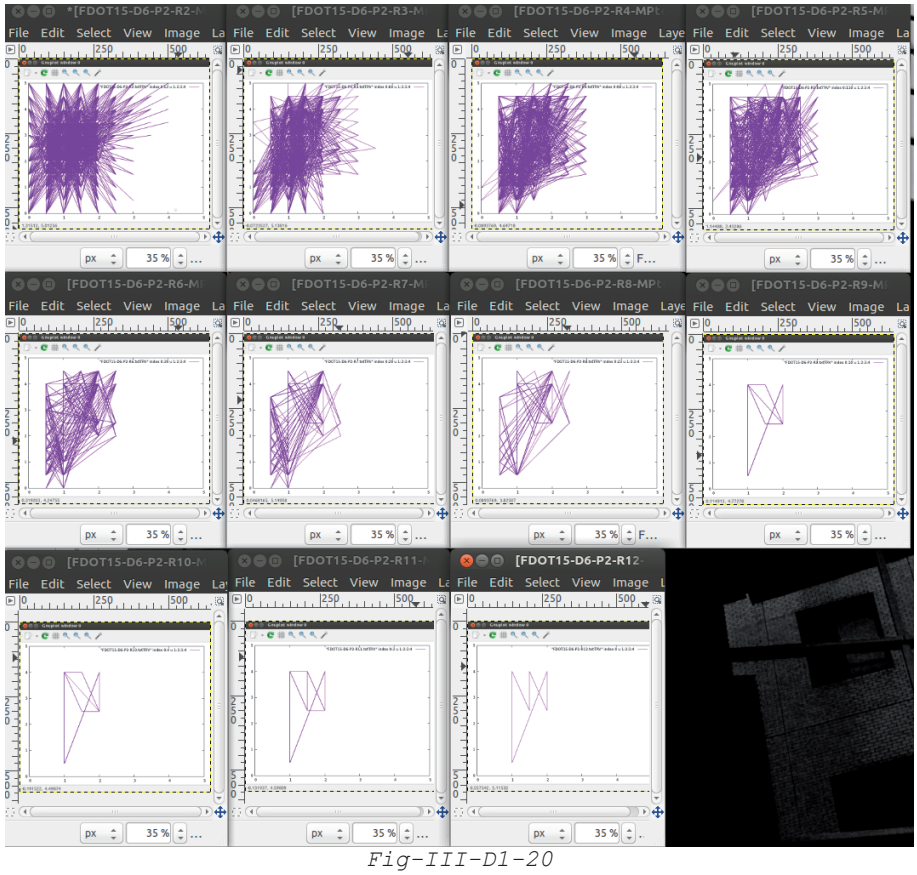


Fig-III-D1-20

Fig-III-D1-17/18. Pour ces valeurs de 'k' les calculs sont trop volumineux avec cent-vingt-et-un points-grille.
Même remarque que pour le motif 'A', ces visuels (cf. les figures *Fig-III-D1-17/18/19/20*) sont une sorte d'identité topologique du motif 'F'.

Sur les figures *Fig-III-D2-21/22/23/24* nous avons les frontières exactes du motif 'A' avec 'k' allant de 2 à 6 pour les *k-partitions* optimales ($CG^*_{20,k}$) correspondantes. Les traits verts représentent les différents points P_i des MM^*_{k-p}, reliés entre eux. Les traits noirs représentent les frontières entre les différentes régions (Reg_i), hachurées en couleur, du MM^*_{k-p} associé.
Pour k=2, du fait de la symétrie du motif 'A', nous avons deux multi-points optimaux MM^*_{2-1} et MM^*_{2-2} avec deux *k-partitions* optimales et donc deux solutions optimales sol_1 et sol_2, telles que $CG(sol_1)=CG(sol_2)=CG^*_{20,2}=18.7409$. Il en est de même pour k=3 ($CG^*_{20,3}=12.7673$). Ceci n'est plus le cas pour k>3.

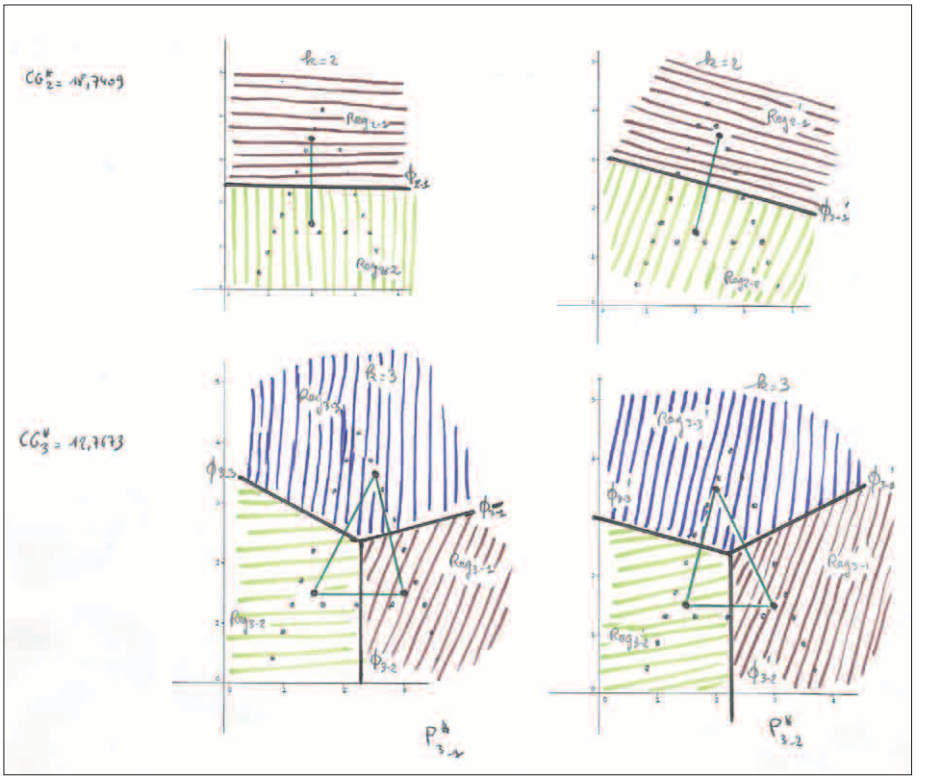


Fig-III-D2-21

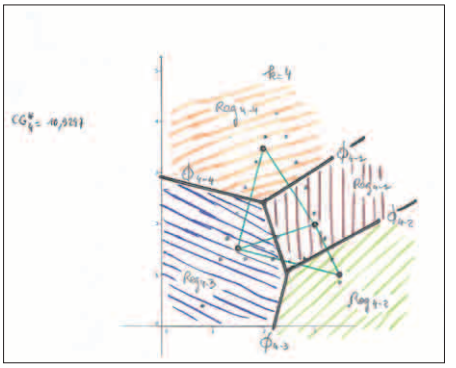


Fig-III-D2-22

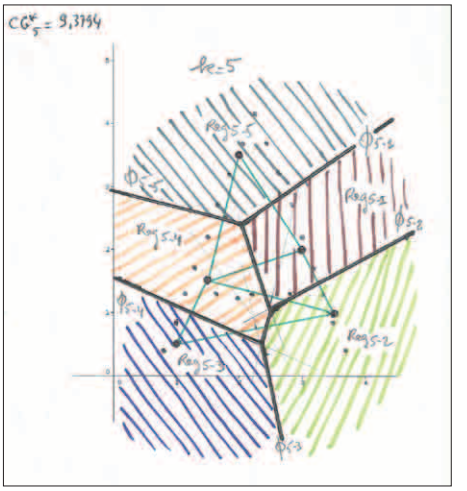


Fig-III-D2-23

DIAGRAMME DE VORONOÏ

En fait les régions Reg_i d'une *k-partition* forment un diagramme de Voronoï, où chacune d'elles, appelée cellule, correspond à l'ensemble des points les plus proches d'un site , par rapport aux autres sites.
Un site, encore appelé germe, correspond à chaque point P_i de MM_{k-p}, considéré ici comme un ensemble 'S' de 'k' points du plan. Le diagramme de Voronoï de l'ensemble de ces sites est une partition du plan en cellules.
La frontière commune à deux cellules s'appelle une arête de Voronoï. Il s'agit d'une portion de la médiatrice de deux points de 'S', ici MM_{k-p}. Les extrémités des arêtes sont les sommets de Voronoï.
Comme le montrent les figures *Fig-III-D2-21/22/23/24*, nous avons les propriétés suivantes[1]:

- chaque cellule (Reg_i) est un polygone convexe (intersection d'ensembles convexes - des demi-plans);

1 CES PROPRIÉTÉS SONT ISSUES DE L'ARTICLE «*MODÉLISATION À L'AIDE DES DIAGRAMMES DE VORONOÏ : RÉSEAU TÉLÉPHONIQUE, ET AUTRES APPLICATIONS.*» S. AUCLAIR-SEMERE, M. JACQUIN, E. LAKRITZ ET A.-M. SANCHEZ.

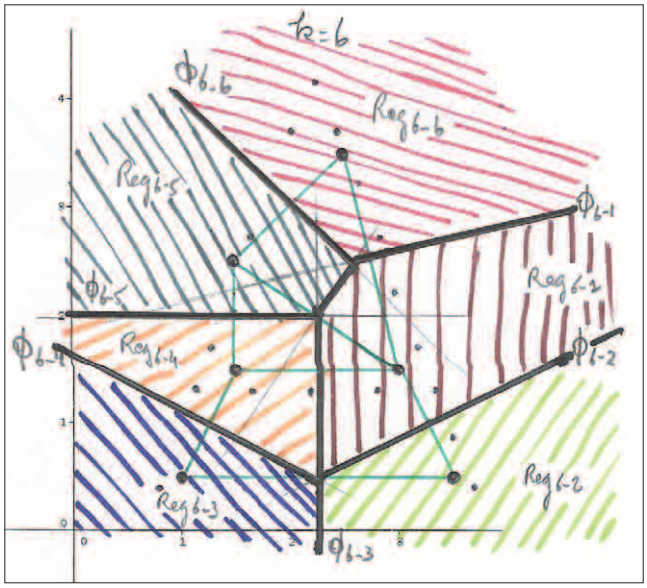


Fig-III-D2-24

- le diagramme de Voronoï de 'k' points partitionne le plan en polygones convexes : chaque polygone contient exactement un point P_i et chaque point d'un polygone est plus près de son point central P_i (germe) que de tout autre ;
- tout sommet de Voronoï est de degré au moins 3. Pour les valeurs 3, 4 et 5 de 'k' nous avons un degré 3. Par contre pour k=6 (cf. la figure *Fig-III-D2-24*) nous avons quatre points voisins qui sont cocycliques (appartiennent au même cercle inscrit) ;
- pour chaque sommet du diagramme de Voronoï, le cercle passant par les points P_i de MM_{k-p}, voisins de ce sommet, ne contient aucun autre point P_j de MM_{k-p} ;
- si on imagine des cercles dont le rayon augmente à la même vitesse, centrés aux points P_j de MM_{k-p}, un point 'x' est atteint par un, deux ou trois cercles selon qu'il est à l'intérieur d'une région, sur une arête ou à un sommet de Voronoï ;
- les centres des grands cercles vides sont alors les sommets du diagramme de Voronoï.

Pavage de Delaunay

Nous obtenons un pavage de Delaunay de MM_{k-p}, en partant du diagramme de Voronoï de MM_{k-p} issu des régions Reg_i, en reliant par une arête (en vert sur les figures *Fig-III-D2-21/22/23/24*) deux points P_i de MM_{k-p} qui sont dans des régions Reg_i distinctes de Voronoï, mais ayant une arête commune.
Ceci équivaut au critère suivant : deux points P_i de MM_{k-p} sont reliés s'il existe un cercle contenant ces deux points sur sa frontière et aucun autre point de MM_{k-p} sur sa frontière ou dans son intérieur.

On obtient alors les résultats suivants :

- on appelle triangulation un graphe planaire auquel on ne peut ajouter d'arête dans couper une arête existante. Le pavage de Delaunay de MM_{k-p} est une triangulation s'il n'y a jamais quatre points de MM_{k-p} cocycliques. C'est le cas pour k=3, k=4 et k=5 (cf. les figures *Fig-III-D2-21/22/23*), mais pas pour k=6 (*Fig-III-D2-24*) ;
- s'il n'y a jamais quatre points cocycliques dans MM_{k-p}, le dual du diagramme de Voronoï est la triangulation de delaunay de MM_{k-p} ;
- trois points de MM_{k-p} donnent un triangle de Delaunay si leur cercle circonscrit ne contient aucun point de MM_{k-p} en son intérieur ;
- la triangulation de Delaunay est parmi toutes les triangulations de MM_{k-p} celle qui maximise l'angle minimum de tous les triangles.

Nous faisons apparaître en vert, sur les figures *Fig-III-D2-21/22/23*, les triangulations de Delaunay associées à MM_{3-p}, MM_{4-p} et MM_{5-p}.

Descentes de gradients locaux

Lors des heuristiques résolutives, tant que chaque point P_i d'un multi-points MM_{k-p} donné, reste dans sa région, alors les différents MM_{k-p} générés restent dans la *k-partition* associée. Cela permet de suivre le gradient d'optimalité propre à cette *k-partition*.

Les figures *Fig-III-D2-25/26/27/28/29/30* visualisent les gradients d'optimalité locaux à chaque région Reg_i pour k=6.
On a une idée de la façon dont chaque point P_i du MM_{k-p} courant peut se déplacer à l'intérieur de sa région, pour chercher la valeur optimale locale (CG_{min}) à la *k-partition* courante.
Pour compléter cette étude notons que si nous avons autant de régions ('k') que de points-motif ('n') il suffit de mettre chaque point P_i du MM_{k-p} le plus proche possible de

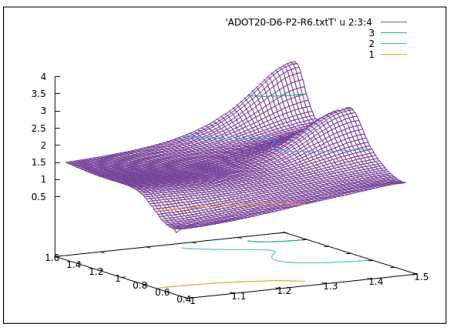


Fig-III-D2-25

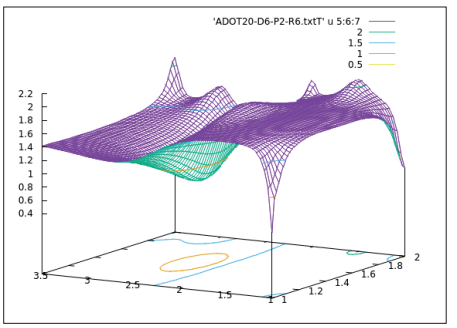


Fig-III-D2-26

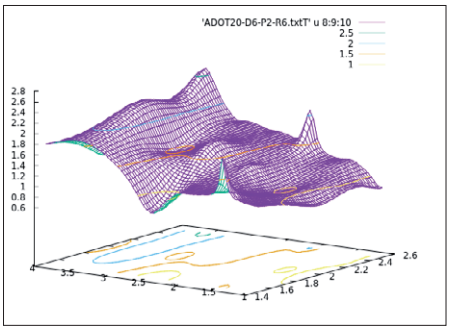


Fig-III-D2-27

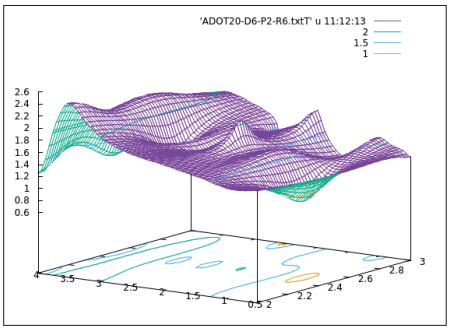


Fig-III-D2-28

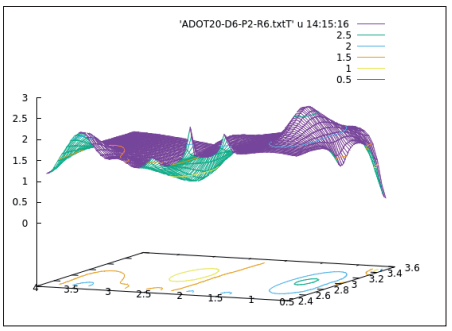


Fig-III-D2-29

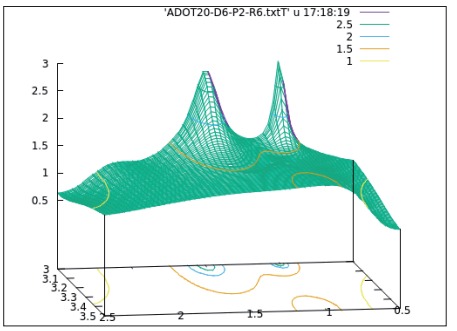


Fig-III-D2-30

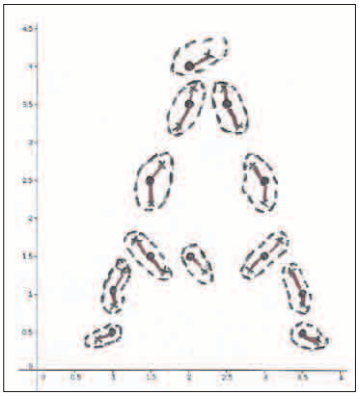


Fig-III-D2-31

l'un des points-motif du motif étudié pour avoir une solution sol_p telle que $CG(sol_p)$ tende vers 0. Cela dépendra de la *maille* de la grille.
Nous avons vu que la *maille* de la grille donnait une limite à cette solution sol_p. Dans notre exemple 'A' avec vingt points-motif sur une grille avec D=6 et Pas=2 (cent-vingt-et-un points) nous obtenons $CG^*_{n=20,k=12}$=5.274 (cf. la figure *Fig-III-D2-31*), la valeur limite étant atteinte avec $CG^*_{n=20,k=16}$=4.9032, c.-à-d. avec k=16.

Nous rappelons également que la frontière entre deux régions Reg_i et Reg_j n'est pas qu'une arête de Voronoï, si l'on considère la notion de faisceau de frontières vue dans la partie III-B3.

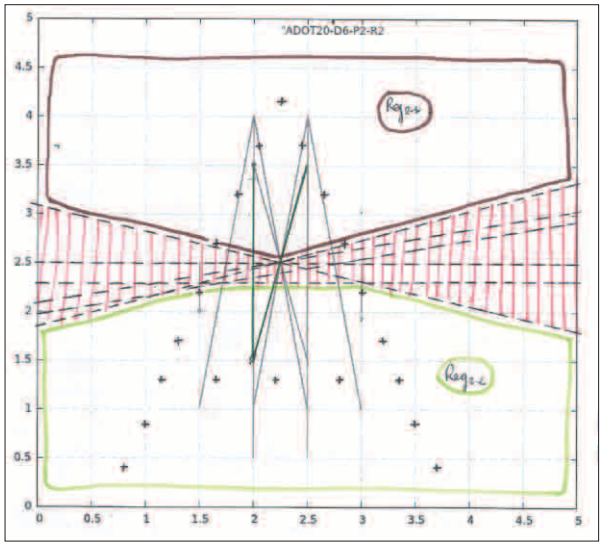


Fig-III-D2-32

En effet il faut considérer l'ensemble des MM_{k-p} qui appartiennent à la *k-partition* en cours. Comme on peut le voir sur la figure *Fig-III-D2-32*, on passe de Reg_i à Reg_i' (Reg_{2-1} en marron) et de Reg_j à Reg_j' (Reg_{2-2} en vert).
Tant que les points P_1 et P_2 des MM_{2-p} de la *k-partition* courante restent respectivement dans Reg_{2-1} et Reg_{2-2}, alors la *k-partition* est préservée (invariant topologique) et la descente de gradient peut opérer.

Pour terminer cette approche nous pouvons regarder sur notre exemple 'A' comment évolue approximativement l'axe du faisceau frontière pour k=2 en fonction des *2-partitions*:

- sur la figure *Fig-III-D2-33* nous avons respectivement l'axe du faisceau de frontières pour la *k-partition* optimale ($Part^*$), puis les trois suivantes ($Part_1$, $Part_2$, et $Part_3$);
- sur la figure *Fig-III-D2-34* nous avons respectivement l'axe du faisceau de frontières pour les quatres *k-partitions* suivantes ($Part_4$, $Part_5$, $Part_6$, et $Part_7$);
- sur la figure *Fig-III-D2-35* nous avons respectivement l'axe du faisceau de frontières pour les quatres *k-partitions* suivantes ($Part_8$, Part9, $Part_{10}$, et $Part_{11}$);
- sur la figure *Fig-III-D2-36* nous avons respectivement l'axe du faisceau de frontières pour les trois *k-partitions* suivantes ($Part_{12}$, $Part_{13}$ et $Part_{14}$).

On peut noter des changements d'axes assez radicaux en angle.

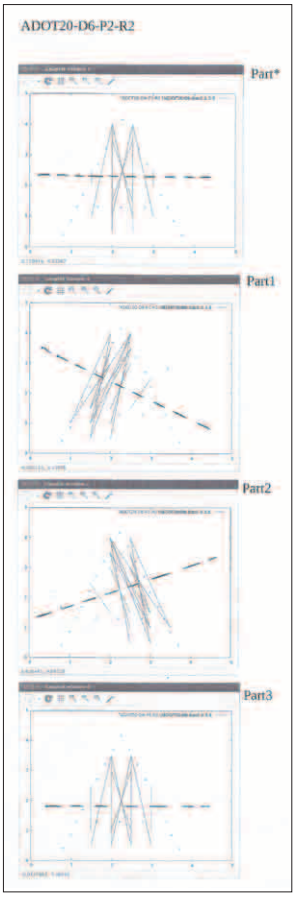


Fig-III-D2-33

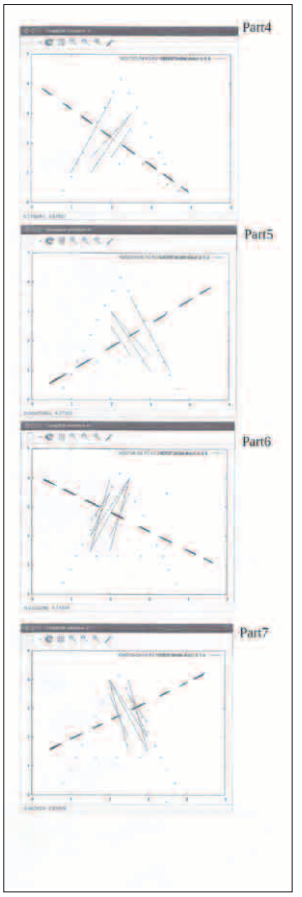


Fig-III-D2-34

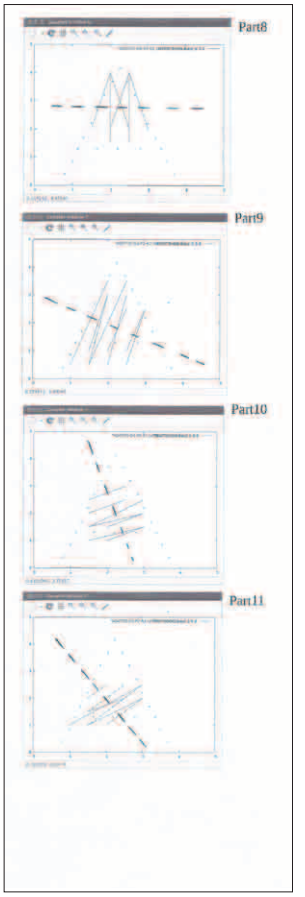


Fig-III-D2-35

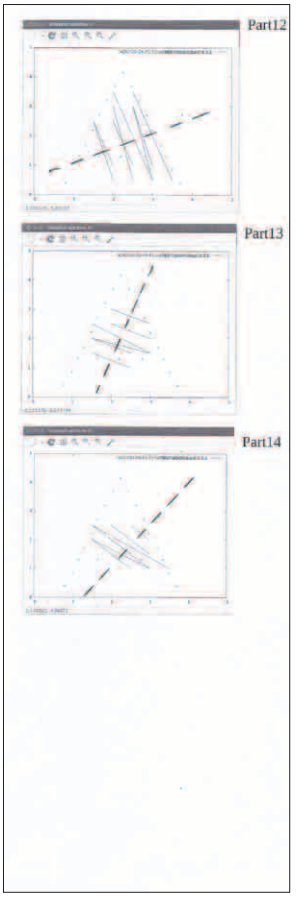


Fig-III-D2-36

En bilan de cette partie nous pouvons dire que les *k-partitions*, composées de multi-points MM_{k-p}, permettent de travailler sur la différenciation entre les motifs étudiés, dans un processus de constitution de modèles. On peut travailler sur la profondeur de l'analyse en jouant sur 'k' (entre 2 et 'n').
D'un point de vue opérationnel (heuristiques résolutives) les *k-partitions* et leurs MM_{k-p} associés, peuvent aussi être vus à travers les diagrammes de Voronoï et/ou leur graphe dual, le pavage de Delaunay. Ces graphes représentent ces *k-partitions* en considérant les faisceaux de frontières liés à leurs différents MM_{k-p} «utiles». Dans certains cas on pourra même utiliser le pavage de Delaunay en tant que triangulation de Delaunay.

ADJACENCE ENTRE MM_{k-p} D'UNE *K-PARTITION*

Nous complétons cette étude par celle de l'adjacence entre les MM_{k-p} «utiles» propres à une *k-partition* $conf_1$ donnée. En effet cela permet de comprendre quels sont les différents chemins permettant de se déplacer de multi-points en multi-points sans changer de *k-partition*.

Dans l'exemple du motif 'A' avec vingt points-motif, un *pas* de 2 et trente points-grille nous avons cent-onze liens d'adjacence entre les vingt-et-un MM_{k-p} «utiles» de Part*. Ces liens sont visibles sur la figure *Fig-III-D2-37*.

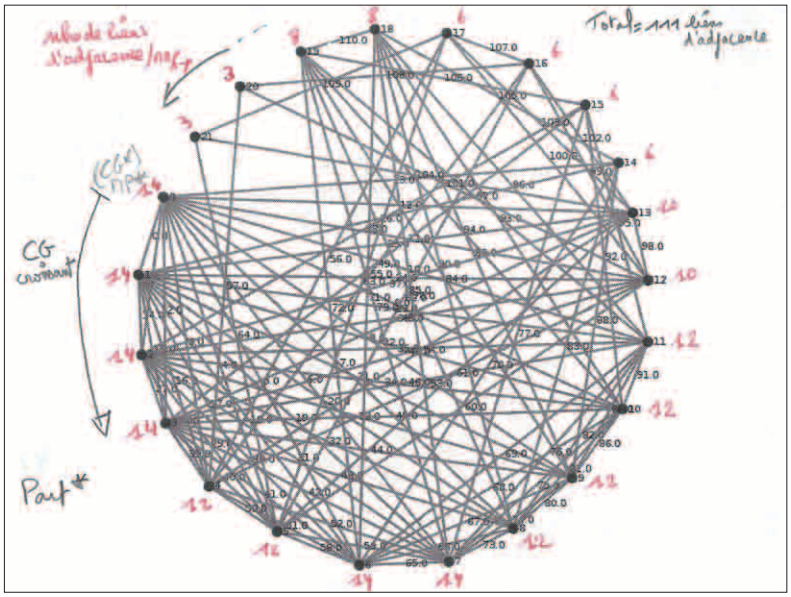


Fig-III-D2-37

En rouge figurent les nombres de liens d'adjacence entre les MM_{k-p}. On constate que quatorze MM_{k-p} sur vingt (70%) sont adjacents avec MM^*_{k-p} (le multi-points optimal), alors que les six restants (30%) sont à une distance ('γ') de 2. Autrement dit il est possible de passer rapidement de n'importe quel multi-points «utile» de $Part^*$ vers le multi-points optimal (CG^*) sans sortir de $Part^*$.

On peut également constater que la connectivité des MM_{k-p} entre eux décroit d'autant que le MM_{k-p} considéré s'éloigne (son CG correspondant augmente) de MM^*_{k-p} (multi-points optimal).

Sur les figures *Fig-III-D2-38/39/40* nous avons les adjacences des MM_{k-p} dans les *k-partitions* $Part_1$ (soixante-et-un liens d'adjacence au total), $Part_2$ (soixante-et-un liens d'adjacence au total) et $Part_3$ (cent-dix-neuf liens d'adjacence au total). Nous constatons la baisse de connectivité au fur et à mesure que l'on s'éloigne du MM_{k-p} optimal dans chaque *k-partition*.

On peut interpréter ceci en disant que la probabilité de passer d'un MM_{k-p} quelconque à MM^*_{k-p} augmente au fur et à mesure que l'on se rapproche de MM^*_{k-p}. La distance ('γ') maximale entre deux MM_{k-p} «utiles» d'une *k-partition* est inférieure ou égale à 2.

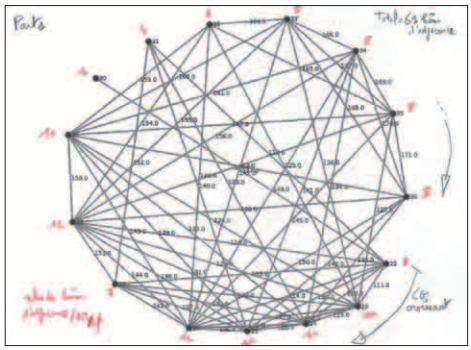


Fig-III-D2-38

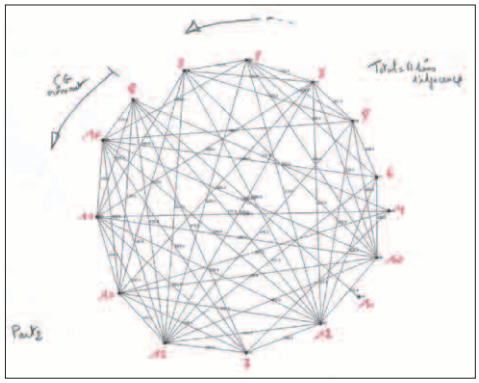


Fig-III-D2-39

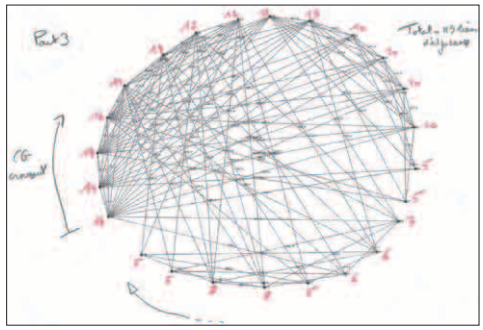


Fig-III-D2-40

III-D3) Étude de l'adjacence entre *k-partitions*

L'étude de l'adjacence entre deux *k-partitions* $conf_1$ et $conf_2$ distinctes, consiste à regarder quels sont les multi-points MM_{k-i} et MM_{k-j}, respectivement appartenant à $conf_1$ et $conf_2$, qui sont voisins ou adjacents, c.-à-d. à une distance $\gamma(MM_{k-i},MM_{k-j})=1$ l'un de l'autre (cf. la partie III-B3, sur la figure *Fig-III-B3-17*). S'il en existe, alors les deux *k-partitions* sont voisines/adjacentes ($\delta(conf_1,conf_2)=1$, cf. la partie III-B4, sur la figure *Fig-III-B4-28*).

Toujours avec l'exemple du motif 'A' avec vingt points-motif, un *pas* de 2 et trente points-grille nous obtenons (cf. la figure *Fig-III-D3-41*) cent-quatre liens d'adjacence entre $conf^*_1$ ($Part^*$) et $conf_1$ ($Part_1$, la *k-partition* suivante, par CG^* croissant). Dans chaque *k-partition* les MM_{k-p} sont rangés de droite à gauche en fonction de leur $CG(sol_p)$ associé (du plus faible à droite vers le plus grand vers la gauche).

En première analyse nous observons que les MM_{k-p} les plus intéressants dans chaque *k-partitions* sont parmi les plus connectés en nombre de liens (neuf pour MM^*_{k-p} de $Part^*$ et douze pour MM^*_{k-p} de $Part_1$).

Les deux *k-partitions* sont adjacentes par une grande variété de chemins (cent-quatre en tout). On peut interpréter cela en disant qu'il existe de nombreux chemins pour passer de $Part^*$ à $Part_1$, quelque soit le MM_{k-p} «utile» de départ de l'une ou l'autre des *k-partitions* adjacentes.

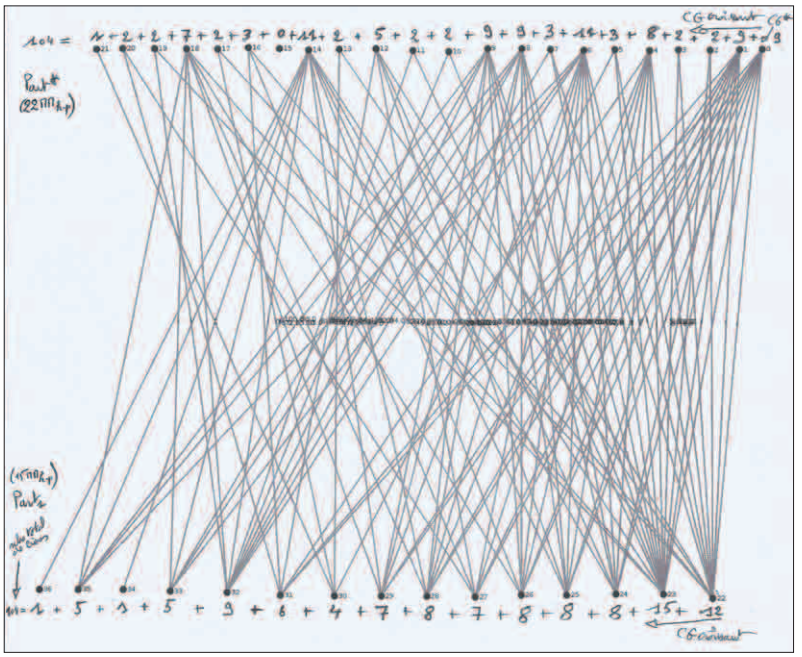


Fig-III-D3-41

Si l'on regarde, sur la figure *Fig-III-D3-42*, l'adjacence entre $Part^*$ et $Part_2$ (la *k-partition* suivant $Part_1$ par valeur croissante de CG^*) nous obtenons encore une forte capacité d'adjacence entre elles, puisque nous avons cent-trois liens d'adjacence.

Nous avons aussi la même propriété de forte connexion pour les MM_{k-p} qui tendent vers MM^*_{k-p}, avec quand même seulement deux liens pour MM^*_{k-p} de $Part^*$ et deux liens également pour le MM_{k-p} suivant de $Part^*$.

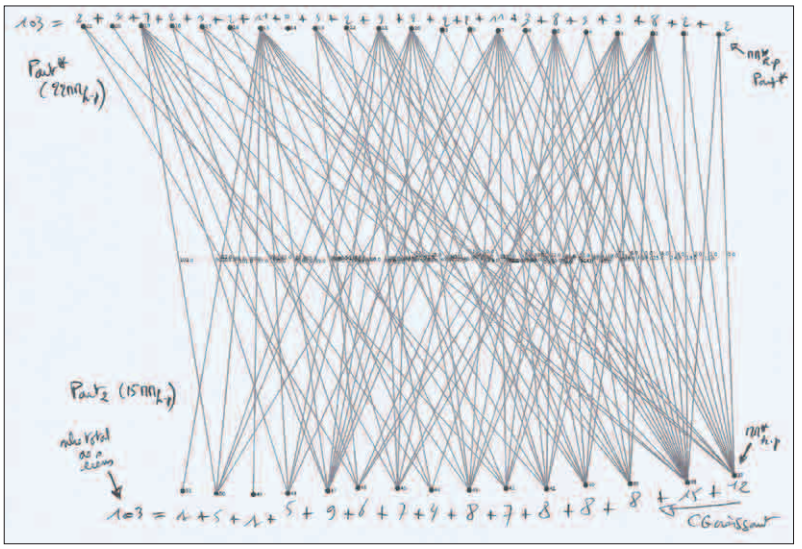


Fig-III-D3-42

Sur la figure *Fig-III-D3-43* nous avons le nombre de liens d'adjacence entre $Part^*$ et toutes les autres *k-partitions* «utiles». Au centre de la figure nous avons $Part^*$ qui est représentée par 0. Chaque segment relie $Part^*$ avec une *k-partition* «utile» dont le numéro figure à l'extrémité du lien, la valeur en rouge représente le nombre de liens d'adjacence entre ces deux *k-partitions*. Par exemple $Part^*$ est reliée avec $Part_{57}$ au sud-est de la figure *Fig-III-D3-43*, avec cinq liens d'adjacence.

La longueur des segments représente approximativement l'inverse du nombre de liens d'adjacences (passages) entre deux *k-partitions*.

Dans cet exemple on dénombre soixante-quatre *k-partitions*, dont $Part^*$, trois-cent-trente-cinq MM_{k-p} et huit-cent-cinquante-sept liens d'adjacence entre elles (cf. la figure *Fig-III-D3-44*).

Trente-six *k-partitions* sont directement reliées (distance $\delta(conf_1, conf_2)=1$) par des liens d'adjacence à Part* (57%) et vingt-sept ne le sont pas, mais le sont toutes (les vingt-sept) indirectement (distance $\delta(conf_1, conf_2)=2$).

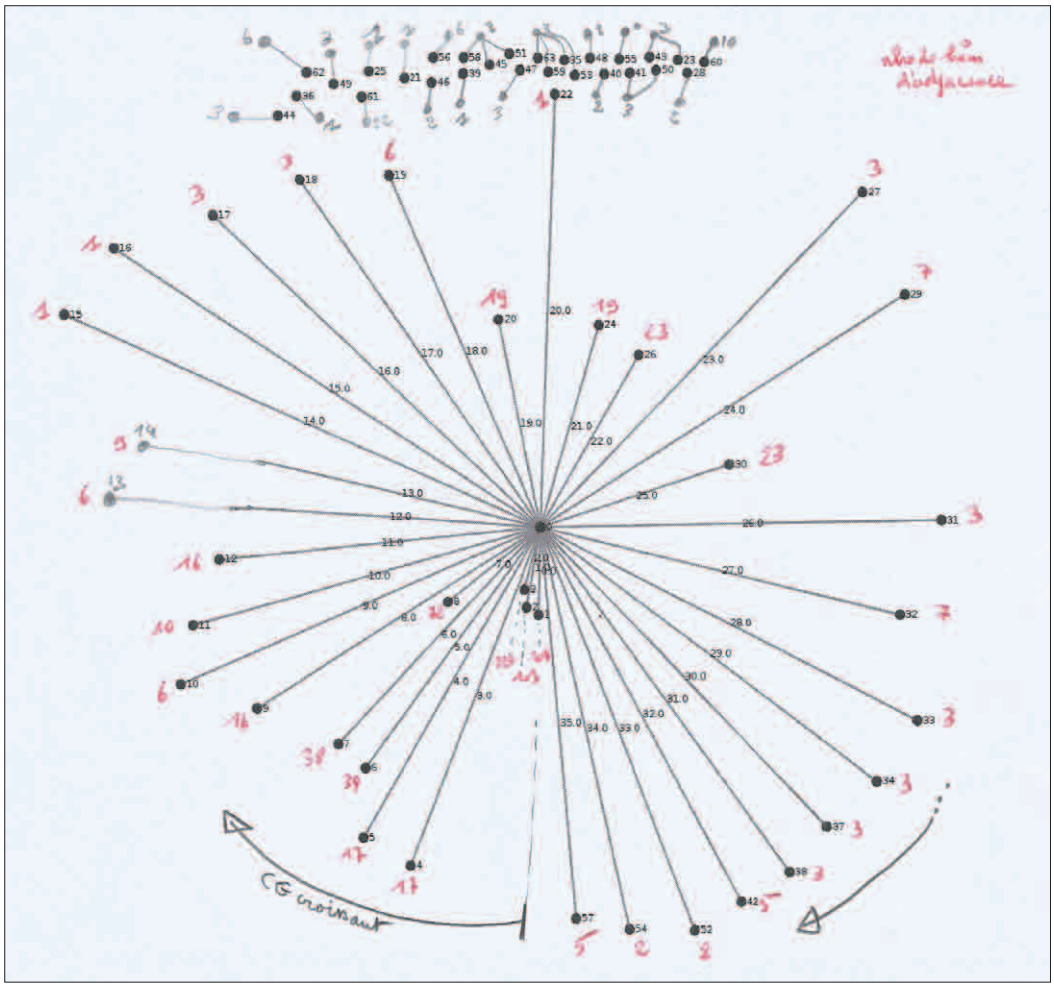


Fig-III-D3-43

Nous avons représenté ces vingt-sept *k-partitions* sur la partie haute de la figure *Fig-III-D3-43* en les reliant (crayon à papier) avec l'une des *k-partitions* qui leur est adjacente et qui est elle-même adjacente avec Part*.

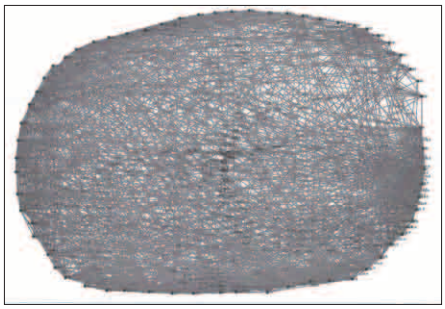


Fig-III-D3-44

La figure *Fig-III-D3-43* montre clairement que quelques *k-partitions* ($Part_1$, $Part_2$, $Part_3$ et $Part_8$) représentent une majorité (55,59%) des liens d'adjacence vers Part*. Cela signifie que peu de *k-partitions* sont très fortement connectées à Part*.

Une autre façon de voir le passage d'une *k-partition* à une autre vers Part* consiste à prendre un point fixe appartenant à un MM_{k-p} de Part*, sur la figure *Fig-III-D3-45* le point (x=2,y=1), et à regarder dans toutes les *k-partitions* «utiles» si l'un des points des MM_{k-p} correspond à ce point fixe. On pourra ainsi en déduire un graphe (cf. la figure *Fig-III-D3-46*) d'adjacence entre ces *k-partitions* autour de ce point fixe. Sur la figure *Fig-III-D3-45* les quinze MM_{k-p}, des différentes *k-partitions* qui possèdent ce point fixe, sont regroupés par *k-partitions*.

Par exemple nous avons $Part_3$ avec les MM_{k-p} 52, 53 et 56.

Le graphe induit (cf. la figure *Fig-III-D3-46*) montre les nombreux chemins entre les *k-partitions* pour aller vers $Part^*$ en partant du point fixe (x=2,y=1) et en déplaçant le deuxième point du bi-points associé.
Nous voyons sur cet exemple qu'il existe des chemins pour rejoindre $Part^*$ sans nécessairement déplacer simultanément tous les points d'un MM_{k-p}.

À travers l'étude des calculs exhaustifs de l'ensemble des *k-partitions* d'un motif nous arrivons à la construction des graphes d'adjacence entre les multi-points d'une même *k-partition*, et de ceux entre des multi-points entre des *k-partitions*.
Pour une lettre comme 'A' avec trente points-grille on observe une très grande connexité des graphes. Cela traduit les innombrables chemins possibles pour parcourir les multi-points (MM_{k-p}) liés à ce motif.

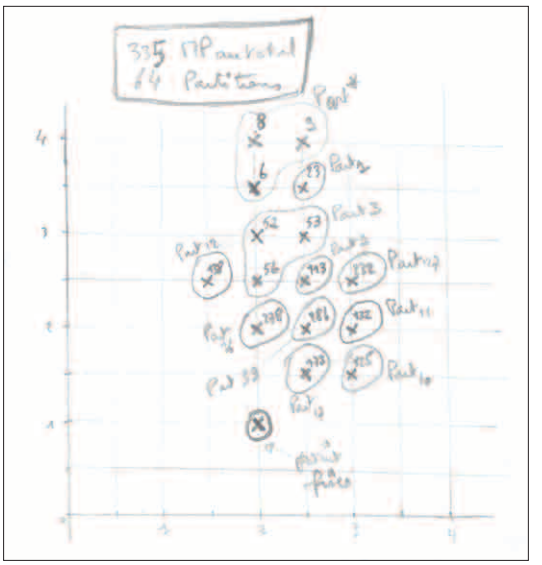


Fig-III-D3-45

L'étude exhaustive s'arrête provisoirement là. Elle reprendra après la partie sur les simulations, basées sur des heuristiques résolutives. Elles vont permettre, par des processus d'exploration, de parcourir implicitement ces graphes extrêmement connectés. En y introduisant des mécanismes d'apprentissage/habituation, comme *la dérive des connaissances*, ces explorations successives vont mettre en évidence des chemins privilégiés entre les multi-points qui seront la réelle empreinte du motif étudié.

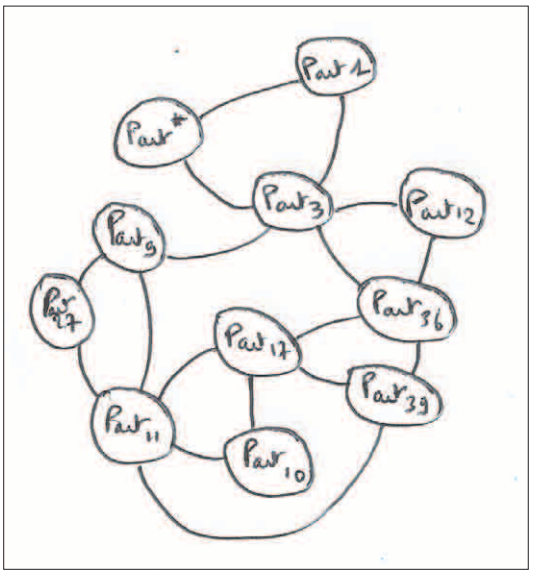


Fig-III-D3-46

L'important n'est pas tant de trouver le multi-points optimal (MM^*, puits de $Part^*$), mais le ou les chemins nécessaires pour s'en approcher. Ce sont ces chemins qui caratérisent le motif. C'est donc dans ces chemins qu'il faudra chercher la différenciation entre les motifs.

HEURISTIQUES RÉSOLUTIVES POUR LE PARTAGE DE RESSOURCES $C_N S_K$

IV

Un système complexe intègre son environnement de façon intime et sous différentes facettes. L'un des moyens de s'en persuader est de construire artificiellement des systèmes complexes et d'étudier où est effectivement la limite, si elle existe, entre l'intérieur et l'extérieur du système, là où commence son environnement.

Nous avons vu, au chapitre III, que la résolution exhaustive du système C_nS_k devient calculatoirement très vite inaccessible, et ce d'autant plus que l'on ne tient pas compte du contexte. Cela interroge sur le fait que l'environnement du système fait partie intégrante du système complexe, et qu'une modélisation mathématique négligeant cette dimension du problème introduit un biais dont la déformation est telle, que le système étudié peut s'en trouver complètement dénaturé.

Cette partie propose une approche heuristique pour la résolution de problèmes de type C_nS_k, qui se fonde sur une intégration de l'environnement tout au long du processus auto-organisé de résolution. Il s'agit de conduire le système dans un état global émergent et acceptable.

De façon implicite, nous nous référons au concept d'autopoïése de F. Varela et H. Maturana, qui nous paraît tout-à-fait pertinent ici, pour expliquer et comprendre comment un ensemble d'acteurs du système, autonomes, obéissant à leur propre fonctionnement interne, sont capables à un moment donné d'agir collectivement de telle sorte que le système passe d'un état d'instabilité à un état de stabilité recherché. Nous montrons comment l'environnement, au-delà de fournir un cadre spatio-temporel pour l'interaction entre les membres du système, sert de support à un processus de synchronisation qui conduit le système dans ce que nous appelons une *interaction efficiente*. Cette *interaction efficiente* est le résultat de processus décisionnels décentralisés, mais synchronisés, s'appuyant, entre autres, sur la notion de période de mémorisation de l'historique des actions menées dans l'environnement. Elle débouche sur la perception d'un comportement global émergent.

Introduction

IV-A

Nous reprenons l'exemple simple, dans sa description, du système complexe C_nS_k, vu au chapitre III, qui est illustré par un service de traduction automatique de documents. La complexité provient du fait qu'un grand nombre 'n' de clients C_i possèdent des besoins changeants et imprévisibles en matière de traduction de documents. Afin de ne pas être tributaire des aléas du réseau et de faire face à des situations changeantes au meilleur coût, le prestataire du service imagine de disposer un nombre variable de serveurs S_j judicieusement placés et déplacés sur des nœuds nd_l du réseau. Les serveurs anticipent leurs besoins futurs en ressource auprès des nœuds en fonction des demandes passées.

On peut aborder ce problème en considérant une grille de (DxD) positions sur laquelle pourront se déplacer les serveurs. On peut aussi considérer, comme nous le ferons dans les simulations, que les serveurs peuvent se déplacer dans un espace continu (liaisons satellites par exemple). Nous faisons l'hypothèse réaliste que les clients sont considérés comme étant immobiles (sur des points fixes). Le côté de plus en plus nomade des clients pourrait être pris en considération dans nos heuristiques résolutives. Toutefois cela conduirait à une complexité supérieure non souhaitée pour cette première approche du problème, même si les heuristiques décentralisées et auto-organisées pourraient parfaitement l'intégrer.

À partir de là, optimiser notre système dans sa dimension la plus triviale, consiste simplement à trouver dans le temps, les meilleures positions de nos serveurs dans le réseau, afin de satisfaire les demandes changeantes des clients, tout en minimisant le coût de la prestation (minimisation des échanges).
D'un point de vue théorique, au temps 't' il suffit de passer en revue toutes les possibilités de placement des serveurs et de calculer pour chacune d'elles, la distance (pondérée ou non) totale entre les clients et cette position.

Nous avons vu au chapitre III que les calculs au temps 't' pouvaient être trop longs et restaient approximatifs, car nécessairement discrétisés sur une grille. En outre ils sont très sensibles au nombre de serveurs déployés et au nombre de clients en jeu.
Notre proposition consiste à expliciter la notion d'environnement dans les processus de résolution, en nous basant sur des entités autonomes (agents) qui interagissent et coopèrent entre elles via et dans cet environnement.

Environnement et interaction efficiente

L'environnement est un cadre spatio-temporel partagé dans lequel évoluent conjointement des entités susceptibles de participer à l'apparition de phénomènes «collectifs». Ces entités sont autonomes dans la mesure où elles doivent conduire en même temps leur stratégie comportementale propre. L'environnement joue le rôle de médiateur pour les interactions. Ces dernières peuvent être directes, ou indirecte (dépôts de traces, recours à des intermédiaires,…).

La nature n'est pas en reste dans ce domaine. C'est par exemple le cas de la grande douve du foie qui modifie le comportement des fourmis afin qu'elles soient plus facilement ingérées par les moutons[1].

Afin que des entités autonomes interagissent, il est nécessaire qu'elles se perçoivent les unes les autres dans l'environnement partagé; mais cela n'est pas suffisant pour produire ce que nous appellerons une *interaction efficiente*, c.-à-d. productrice de comportements émergents.

interaction efficiente
comportement émergent

Dans l'exemple vécu «de la voiture en panne», seul vous n'arrivez pas à dégager votre voiture. À plusieurs non plus si les efforts de chacun ne sont pas synchronisés. Cette synchronisation est loin d'être évidente à mettre en œuvre dans les comportements émergents. Il faut déjà savoir sur quoi elle opère.

Dans nos expérimentations nous avons décomposé l'interaction en deux parties distinctes, mais qui fonctionnent ensemble. On trouve l'interaction basique (par exemple celle qui consiste à pousser à plusieurs une voiture dans une direction donnée), et celle (efficiente) qui a pour but de synchroniser les entités en jeu (c'est celle qui consiste à faire en sorte que tout le monde pousse en même temps), afin d'obtenir le comportement émergent souhaité.

Clarifions ce que nous avons observé dans nos expériences autour de ce concept de synchronisation efficiente. Cette synchronisation ne se suffit pas d'un temps partagé (être présents ensemble/connectés à un moment précis).
Notre hypothèse se fonde sur l'idée que le liant qui construit la membrane (au sens autopoïétique), fusse-t-elle virtuelle, est le résultat de l'interaction efficiente. Ce niveau membranaire atteint, l'interaction basique (comportement de base des entités) reste inchangée, mais dans la mesure où l'environnement ayant été localement synchronisé, de nouveaux comportements collectifs peuvent émerger.

Pour fonctionner, nos agents sont dotés d'un mécanisme très simple d'apprentissage, basé sur la faculté de mémorisation des actions antérieures, et d'une perception des effets engendrés par ces actions antérieures sur l'environnement (*feedback* de l'environnement).
Cette activité mémorielle donne un certain degré

1 Dans l'exemple de la grande douve du foie et des fourmis, les douves ont élaboré le moyen d'interagir avec l'environnement en passant du mouton au mouton via les escargots puis les fourmis. Elles se développent dans le foie des moutons, y pondent des œufs, qui ne peuvent y éclore. Les œufs quittent le mouton via ses excréments, puis les larves écloses seront récupérées par des escargots, où elles se multiplieront avant d'être à nouveau rejetées dans les mucosités de l'escargot qu'il crache en période de pluie. Ces mucosités attirent les fourmis, qui les mangent. Les larves s'y développent, jusqu'au moment où il est temps pour elle de retrouver le foie d'un mouton, où elles pourront se reproduire. Les larves prennent le contrôle des fourmis pour les faire monter au sommet des herbes susceptibles d'être broutées par les moutons. Inutile de dire que l'environnement conditionne toutes ces stratégies. En particulier les herbes choisies par la larve pour y conduire la fourmis doit être appréciée des moutons (aspect spatial de l'environnement). D'autre part il faut y être aux heures fraîches, moment pendant lequel les moutons broutent (aspect temporel de l'environnement). Malheureusement pour les larves, les fourmis quittent leur nid aux heures chaudes pour circuler au pied des herbes. Toute cette folle stratégie d'interaction que mettent en œuvre les larves afin de se développer, ne pourrait pas fonctionner sans un moyen de synchroniser les différents intervenants dans la chaîne d'interaction. Le moyen extraordinaire trouvé par les larves est de piloter le cerveau de la fourmi, afin de la synchroniser sur le fonctionnement des moutons, via l'environnement.

d'autonomie à nos entités, qui se traduit par un processus décisionnel simple et efficace, leur permettant de s'ajuster continuellement à la perception qu'elles ont de l'environnement.

Plus concrètement nos entités artéfactuelles sont, par exemple, des serveurs de traduction automatique de documents, dupliqués et répartis dans le réseau, afin de satisfaire au mieux les besoins des clients, tout en optimisant les coûts (communication, réservation de ressources de calcul sur des machines,…) et en garantissant une qualité du service (délais, contenu,…).
Pour cela ils peuvent se regrouper localement afin de constituer une sorte de membrane leur permettant ainsi de faire face collectivement à une demande particulière (un ensemble de demandes concomitantes) nécessitant une coopération entre différents serveurs.
Leur apprentissage du feedback de leur interaction basique avec l'environnement est ici réduit à la quantité de ressources informatiques (temps processeur de la machine où ils fonctionnent) nécessaire à la réalisation des demandes qu'ils reçoivent via l'environnement. On pourrait l'enrichir, mais cela suffit pour montrer le principe de synchronisation efficiente.
Leur comportement basique consiste donc à recevoir des demandes et à les satisfaire dans la mesure du possible (en terme de quantité de ressources allouées). L'apprentissage consiste alors à mémoriser sur une période donnée les demandes reçues en corrélation avec les ressources allouées dans l'environnement. Le processus de décision interne repose sur la capacité de l'entité à maintenir dans le temps une adéquation optimale entre les ressources qu'elle alloue et la réalisation des demandes qu'elle reçoit.

À l'image des approches markoviennes qui consistent à ne pas tenir compte d'un historique trop lointain au risque de prendre des décisions infondées pour l'avenir proche, le problème de nos serveurs de traduction est de définir une bonne fenêtre de mémorisation (empan) afin d'avoir un processus décisionnel efficace (détection fine des changements liés à l'environnement, que ce soit les demandes des clients ou les pannes des machines).

Ce problème se complique dans la mesure où très souvent l'optimisation globale du système (ensemble du service de traduction) ne peut être assurée que par l'addition du comportement des serveurs pris indépendamment les uns des autres, mais doit les faire coopérer. Les politiques décisionnelles internes à chaque serveur doivent coopérer.

Nos expérimentations montrent comment cette coopération, conduisant à une sorte de mécanisme décisionnel collectif complètement décentralisé, repose en fait, entre autres choses, sur une synchronisation des historiques (période de mémorisation) des différents agents. Nous observons l'émergence du comportement collectif attendu (ensemble de décisions locales conduisant le système durablement dans l'état d'équilibre global entre les ressources allouées et les ressources nécessaires), uniquement lorsque une certaine période de mémorisation commune est atteinte.

Pour cela chaque agent dispose au départ d'une période de mémorisation initialisée à 1; ce qui correspond à un comportement réactif sans mémorisation des expériences passées.

Nous avons doté les agents d'un protocole leur permettant d'échanger entre eux leur période de mémorisation. Il est fondé sur l'idée qu'un agent peut localement percevoir qu'il est dans un état «stable» ou «instable», en se basant sur des grandeurs qui lui sont propres, qui le caractérisent et qui lui sont accessibles. Nous nous appuyons sur deux grandeurs principales que sont leur mobilité et la variabilité de leurs besoins. Chaque agent considèrera qu'il est dans un état stable s'il ne se déplace plus et que sa variabilité de besoins est nulle. Dans cet état l'agent cherche à propager aux autres agents avec qui il coopère, sa propre période de mémorisation.

En observant le système évoluer, on voit qu'il converge vers un état global stable. Il entre ainsi dans le régime d'interaction efficiente, ce moment où tous les agents impliqués utilisent des périodes communes de mémorisation (comportement émergent). Ils se sont synchronisés.

Le système étant très fortement dynamique, il explore l'espace des états des périodes de mémorisation, et n'arrive pas nécessairement à se stabiliser directement. Les courbes montrent très nettement qu'au pire des cas il tourne autour de cet état d'équilibre. Cet espace définit donc les différentes périodes de mémorisation possibles. Il faut se rappeler qu'une période de mémorisation correspond à la prise en compte dans le processus décisionnel courant des expériences actions/perceptions des agents sur l'environnement pour une période allant de l'instant présent à un passé choisi.
Autrement dit la synchronisation, conduisant à l'interaction efficiente, repose sur un partage entre les différents agents d'un espace de mémorisation commun, utilisé dans leur processus décisionnel interne.

Tout comme pour les changements climatiques, pour lesquels il faut connaître la bonne période d'observation des phénomènes, pour leur donner du crédit, nous avons également constaté que pour un système donné, la convergence se fait toujours autour d'une valeur précise de la période de mémorisation qui serait, dans ce cas, la valeur optimale.

Pour revenir sur le caractère autopoïétique du dispositif, et le rôle de l'environnement dans ce dernier, soulignons le fait que l'ensemble des agents conserveront toujours leur comportement propre, avant, pendant et après l'équilibre global atteint.

Grâce aux interactions qu'ils ont entre eux, via l'environnement, ils explorent collectivement l'espace des états de la période de mémorisation sur laquelle ils se basent pour mettre en œuvre leur processus décisionnel interne.
Ceci correspond à la phase de construction de la membrane, ou d'auto-organisation.

Enfin à l'équilibre trouvé, la membrane virtuelle est bien en place et fonctionne de telle sorte que le système est stable, robuste aux petites variations et efficient: on peut observer, d'un point de vue extérieur aux agents, le comportement émergent attendu.

53 Je profite de cette occasion, où je mentionne le S.E.P.T., *COOL/CIDRE/CORBA*… et les travaux sur les architectures à objets balbutiantes (fin des années 80), auxquels j'ai participé, pour saluer Jean-Marc Deshayes, un compagnon de route de la première heure. En 1987 j'ai rejoins le S.E.P.T. à Caen, où m'attendait Jean-Marc, jeune ingénieur récemment embauché, pour démarrer des travaux sur les applications bureautiques réparties. Nous nous retrouvâmes tous les deux dans un bureau avec, devant nous, une injonction hiérarchique « faites une application bureautique répartie novatrice ! » et un vide, il faut bien le dire, assez sidéral… Après quelques errances, en particulier en direction des travaux en intelligence artificielle distribuée, mais aussi des travaux commencés au S.E.P.T. sur le sujet par un certain J.-B. Stéfani (Stéfani, J.-B. : *Bureautique : un tour d'horizon*. Document technique interne SEPT/CNET, juillet 1986), avec Jean-Marc nous avons rédigé une note technique (Deshayes, J.-M., Bourdon, F. : « *Interface programmatique d'une couche de communication pour un système bureautique* ». S.E.P.T. document interne, 1988) dont le contenu était une description fonctionnelle du futur sous-système de communication à objets *COOL-v1*. Ce document, d'une vingtaine de pages, a été pris au pied de la lettre par la société Chorus-Systèmes qui, en s'appuyant sur les travaux de thèse de Sabine Habert (S., Habert : « Gestion d'objets et migration dans les systèmes répartis » Paris, thèse de doctorat, université Paris-6 Pierre-et-Marie-Curie, décembre 1989) a développé la couche de communication à objets *COOL-v1* (S., Habert, L., Mosserie, V. Abrossimov : « COOL : Kernel Support for Object-Oriented Environments », Rapport technique n°1211, Rocquencourt, INRIA, avril 1990), puis *COOL-v2* et enfin *COOL-ORB* un produit commercial conforme au standard *CORBA*, proposé par l'OMG. Il faut dire que dans ces années extraordinaires d'audace et de confiance envers les jeunes ingénieurs que nous étions, notre hiérarchie n'avait pas mégoté sur les moyens financiers pour cette opération… Avec Jean-Marc, puis d'autres collègues, venus par la suite en renfort, nous avons contribué modestement, mais indéniablement à l'histoire de l'informatique répartie. En 1991 le standard OMG/*CORBA* apparaissait. Il réunissait, à l'époque, tous les acteurs mondiaux de l'informatique. *COOL-v1* fut l'un des premiers systèmes de ce type, avant même l'existence du standard à l'OMG (1990).

Les nombreuses expériences de réalisation d'applications réparties (*Cidre*, *RCO*,…) sur des architectures de communication et d'exécution de type *CORBA* ou assimilées, auxquelles j'ai participé dans ma carrière, ont montré que la propriété recherchée de transparence à la localisation, offerte par ces architectures aux programmeurs d'application, pouvait devenir problématique.
En effet le coût des interactions entre les composants d'une application répartie peut jouer sur la logique applicative, dans la mesure où c'est l'utilisateur, au sens large, qui va prendre en charge ce coût consécutivement à l'usage qu'il fait de l'application.
En général (y compris dans le cadre de forfaits où les exceptions existent toujours) le comportement des personnes, lorsqu'elles téléphonent, intègre le coût de la communication induit par le contexte (lieu, durée) correspondant.
Si l'on prend l'exemple de la circulation de documents, dans lequel chaque document dispose, entre autres, d'un schéma de route basé sur le rôle organisationnel des personnes de l'entreprise, sa mise en œuvre peut conduire à des allers-retours inutiles entre des personnes éloignées les unes des autres, engendrant une augmentation des coûts et une diminution des performances.
Bref, une méconnaissance complète de la localisation des entités qui doivent interagir conduit à des comportements non-optimaux, voire abérants.

C'est pour ces raisons que nous avons entrepris, il y a plusieurs années, en collaboration avec une équipe de l'Université de Franche Comté, spécialisée en équilibrage de charge, et la société *Chorus Systèmes*, spécialisé en système micro-noyau temps réel, d'étendre les propriétés fournies par les architectures de type *CORBA*[1], de mécanismes capables de gérer dynamiquement le placement des objets qui interagissent en fonction justement de l'évolution de ces interactions.
Cette gestion dynamique avait pour objet d'optimiser les performances des applications et de rationaliser l'utilisation des ressources des systèmes. La plupart des travaux réalisés dans ce domaine concernent soit la gestion des processus, indépendamment de leurs interactions, soit des applications particulières. L'intérêt de la gestion dynamique de la charge dans les systèmes répartis à objets vient du couplage fort entre l'application et le système : il est possible d'obtenir, au niveau du système, des informations sur la structure de l'application.
En particulier, dans un contexte distribué, il est possible de tenir compte des interactions entre les objets pour affiner le placement, en combinant les techniques de gestion de la charge des ordinateurs, de gestion de cluster de fiabilité des machines (nœuds) et des liens entre elles[2].
Plusieurs implantations réelles et non simulées, ont été

1 Ces travaux ont été financés par le Service d'études communes de la Poste et de France Télécom (S.E.P.T.), situé à Caen et créé en septembre 1983 par Louis Mexandeau, alors ministre des P.T.T. sous François Mitterrand et Pierre Mauroy puis Laurent Fabius.

2 Comme nous l'avons fait avec Hugo Pommier et Benoît Romito dans leur thèse respective sur la thématique du stockage de documents dans des réseaux ouverts de type Internet, en se basant sur les nuées d'oiseaux toujours en mouvement ; ces travaux prenaient en compte des propriétés de clustering basées sur la probabilité pour un ensemble de machines de tomber simultanément en panne…

réalisées sur *COOL-v2*, puis sur *COOL-ORB*. Pour cela, nous avons développé ce que nous appelons le gestionnaire d'objets (GO), capable de capturer en continu les interactions effectuées entre les objets via leurs interfaces (IDL *CORBA*). Ce travail proposait une extension (au niveau des *Object Services*) au modèle ORB proposé par *CORBA*, permettant cette collecte d'informations et sur laquelle pourraient être bâtis des services d'administration et d'observation des objets sur le réseau.
Ces travaux, antérieurs aux *web services*, ont montré l'intérêt et la faisabilité des approches décentralisées dans les réseaux ouverts de type Internet.

La principale limitation actuelle à ces approches pour les rendre opérationnelles, réside dans la nature des infrastructures systèmes et réseaux. C'est le constat que nous avions fait pour le stockage des documents. Le paradigme du stockage était poussé à son paroxysme, puisqu'il n'était pas réalisé dans les nœuds, mais dans les liens de communication des réseaux. Nous avons pu démontrer que cela fonctionnait bien, à condition de revoir le fonctionnement des systèmes pour passer d'un mode actuel d'exécution fondé sur les processus, à un mode d'agents actifs beaucoup plus légers et pouvant être extrêmement nombreux sur une même machine.

IV-B1) Définition du problème

Ces travaux, réalisés sur des infrastructures réelles, nous ont conduit à entamer, en parallèle, des réflexions sur ce qui pourrait émerger lorsque ces infrastructures auraient les caractéristiques attendues, comme le fort parallèlisme d'entités légères… C'est ainsi que nous nous sommes intéressés au problème complexe du partage de ressources de type C_nS_k, que nous avons évoqué au chapitre III.

L'étude proposée consiste à observer le fonctionnement du système global (machines/nœuds et réseaux) dans lequel doit naître un ajustement continu de l'organisation de ses composants (objets/ressources) sur les machines.
Cet ajustement doit concourir à minimiser les coûts de communication entre composants et à utiliser au mieux les capacités de l'infrastructure (communication et traitement) proposée. On pourra par exemple rapprocher des composants en très forte interaction ou encore éviter d'engorger une machine par la venue de composants trop gourmands (CPU,…). Il faut être capable d'observer des tendances significatives pour restructurer pertinemment les informations (composants), c.-à-d. sans prendre en compte les éventuels épiphénomènes (bruit) qui pourraient conduire à des restructurations inutiles. On introduit dans le système une forme d'inertie adaptative.

Nous avons retenu les caractéristiques les plus génériques possibles qui définissent les applications de type placement d'objets/partage de ressources dans les systèmes répartis et ouverts.

Pour avancer dans l'exploration de ces idées, nous avons eu recours à l'outil de simulation *oRis* que nous décrivons dans la partie IV-C.
Plus concrètement, le réseau est structuré en domaines, reliés entre eux par des réseaux distants (WAN). Chaque domaine est constitué en première approximation d'un ensemble de machines reliées entre elles par un réseau local (LAN). L'étude que nous décrivons ici ne prend en compte qu'un seul domaine au sens prédéfini. Bien que les simulations peuvent, dès à présent, intégrer plusieurs domaines interconnectés, une réflexion s'impose pour bien comprendre l'impact de cette ouverture.
En effet l'une des difficultés principales de l'approche par la simulation est de projeter de façon pertinente dans le simulateur, une propriété qui nous intéresse dans le domaine réel étudié. Bien souvent le simulateur va permettre différentes représentations possibles pour cette propriété, or il faudra en choisir une qui n'introduise pas de biais dans les mesures observées par les simulations. Nous reviendrons sur ce problème dans la partie IV-E3.
Nous distinguons les machines des logiciels qui fonctionnent sur ces dernières. Chaque machine dispose d'un potentiel de traitement qui lui permet d'exécuter des serveurs (logiciels).
Les clients sont soit des utilisateurs humains soit des logiciels. Nous faisons l'hypothèse qu'ils sont sur des machines dédiées et par conséquent qu'ils ne sont pas mobiles. Cette restriction se veut plus simplificatrice que correspondant à une réalité, bien que dans certaines situations, nous sommes très près de la réalité. C'est le cas par exemple de l'accès à des serveurs d'information sur le Web, où les clients sont fixes et ce sont les serveurs d'information, par le biais des sites «miroir», qui se déplacent. Les besoins sont:

- un client a des besoins qui peuvent être satisfaits par un ou plusieurs serveurs. Son comportement consiste à lancer des requêtes dans le réseau dans lesquelles il décrit la nature (type et conditions) du service qu'il recherche;
- les serveurs sont des logiciels qui réalisent des services donnés (type) dans des conditions qui tiennent compte à la fois des demandes, mais aussi des capacités offertes par le site sur lequel ils fonctionnent. Ils peuvent être déplacés d'une machine vers une autre.

Toujours par simplification, notre vision d'un réseau local est donc la réunion d'un domaine de production, comprenant l'ensemble des machines susceptibles de recevoir des serveurs, et d'un domaine de demande, comprenant les machines sur lesquels fonctionnent les clients.
L'optimisation des machines ne concerne ici que le domaine de production. Un domaine de production étant caractérisé par les machines qui le composent, il aura une capacité globale de production résultant de la somme des capacités de chacune de ses machines. Là encore pour simplifier l'étude, nous considérons que toutes les machines sont identiques en puissance de production.
Lorsqu'un client a des besoins il cherche, par le biais d'un agent «requête», un serveur logiciel capable et disponible pour lui rendre le service correspondant dans les conditions qu'il a exprimées auprès de son émissaire.

Pour répondre à cette demande, le serveur logiciel utilise les capacités de traitement d'une machine. Pour simuler l'adéquation entre les besoins des clients et l'utilisation des ressources physiques nécessaires pour satisfaire ces besoins, nous avons pris un modèle économique simple. Les serveurs produisent de l'information comme une denrée manufacturée qui peut être stockée en attendant d'être vendue aux clients potentiels. Ils peuvent ainsi anticiper sur les demandes en produisant à l'avance; on dira qu'ils constituent une réserve[3].

Dans ce contexte, deux situations peuvent arriver:

- soit l'ensemble des demandes faites par les clients peuvent globalement être satisfaites par la capacité de production des machines du domaine. Dans ce cas il est pertinent d'essayer de répartir au mieux la production des serveurs sur l'ensemble du domaine;
- soit ces demandes dépassent la capacité totale du domaine, auquel cas toute tentative de répartition au mieux de la production dans le domaine ne sera pas satisfaisante.

L'essentiel de notre étude concerne le premier cas.
Chaque «agent requête» cherche un serveur dont les ressources lui permettront de satisfaire les besoins du client, dont il est l'émissaire. En cas de succès les réserves du serveur en seront d'autant diminuées.
Le comportement élémentaire (à chaque pas d'évolution) des clients est d'émettre des requêtes, alors que celui des serveurs est de produire de l'information ou effectuer des réservations pour constituer des réserves/possibilités de traitement.
Dans nos simulations seuls les serveurs se déplacent. L'interprétation du pas de déplacement reste à faire; il faut le comprendre comme étant le déplacement du serveur d'une machine (nœud) du réseau sur une autre. Ce déplacement prend un certain temps à l'image de celui pris par les requêtes pour se déplacer. Nous étudierons (partie IV-E2) l'impact de ces temps de déplacement sur la résolution du problème.

Objectifs

Les heuristiques consistent à optimiser la gestion des ressources matérielles (réseaux et machines) et logicielles (serveurs), en fonction des besoins exprimés par les différents utilisateurs de ces ressources. Cette optimisation se veut continue dans le temps, c.-à-d. qu'elle s'adapte à l'évolution des besoins et des capacités de traitement (réseaux et machines). Sur l'aspect ressources matérielles, l'optimisation recherchée doit veiller, entre autres choses, à ce que les machines soient «bien»

3 On pourrait facilement construire des serveurs sans réserve, ou encore des serveurs dont le matériel produit aurait une durée de péremption. Cette hypothèse n'est pas inintéressante comme le montre certaines études dont l'objectif est de modéliser justement la pérennité de l'information entre sa production et sa consommation. On peut également considérer non plus la production de quelque chose mais simplement la réservation d'une bande passante et d'une capacité de traitement pour une période à venir sur le nœud incriminé. Cette approche est la plus réaliste et la plus générale.

utilisées, c.-à-d. sans déséquilibre de charge trop important entre elles. C'est pour cette raison que nous utiliserons la mobilité des serveurs entre les machines pour réorganiser le réseau local.
La recherche de l'optimisation de l'utilisation des ressources logicielles doit offrir, aux clients du domaine, une forme d'intégrité fonctionnelle. Il s'agit de garantir dans le temps et dans de bonnes conditions (qualité de service) les accès par les utilisateurs du système aux ressources et services dont ils ont besoin. Cela doit se traduire par une configuration qui s'adapte en permanence aux besoins et aux moyens disponibles. Cela se traduit également, pour les clients, par une optimisation des temps de réponse à leurs requêtes, et pour le fournisseur de service, par une configuration matérielle bien dimensionnée (ni trop grande, ni trop faible) et robuste aux pannes.
L'objectif principal recherché est de mettre en évidence le rôle des mécanismes d'auto-adaptation dans le processus de résolution du problème. Ce sera par exemple le cas des stratégies de déplacement des serveurs.

Les mécanismes utilisés

Nous allons décrire les principaux mécanismes, qui seront illustrés à travers les simulations. Ces mécanismes sont:
- un protocole de recherche des serveurs;
- l'auto-ajustement de la production;
- un mode de fonctionnement basé sur des périodes d'observation (PO);
- un processus de réorganisation.

Ces mécanismes sont introduits indépendamment de l'utilisation ou non du modèle relationnel (cf. le chapitre V). Nous aborderons dans ce chapitre V l'apport potentiel du modèle relationnel aux mécanismes de base, développés dans ce chapitre IV.
L'utilisation conjointe et cohérente de ces mécanismes illustrera une application de l'approche relationnelle à un problème concret.

IV-B2) La recherche de serveurs

L'hypothèse forte qui guide le protocole proposé de recherche des serveurs, est que le domaine est changeant et ouvert. Cela signifie que des serveurs peuvent apparaître et disparaître à tout moment, sans que les clients en soit directement avertis. Cette hypothèse correspond tellement à une nécessité que des standards comme *CORBA* ont mis en place des mécanismes sophistiqués[4] allant dans ce sens.
Pour cela nous avons transformé la notion d'invocation synchrone ou asynchrone d'un client sur un serveur par un «agent» requête autonome.

---

4 L'invocation dynamique d'interface ("*Dynamic Interface Invocation*") permet à un client d'invoquer une méthode sur un serveur qui a été rendu accessible dans le système après la création du client. Cela signifie que dans le code compilé du client, il n'y a pas de trace d'une référence à ce serveur. Contrairement à des applications client/serveur classiques, on rend indépendant l'apparition des clients et des serveurs qui doivent coopérer ensemble.

Le client se décharge complètement sur son agent, de tout mécanisme de recherche/négociation du service voulu. Pour le client, le domaine dans lequel il évolue doit lui assurer une «continuité de service», peu importe le serveur ou le nœud impliqué.

Cette notion de «continuité de service»[5], distancie l'utilisateur et la perception qu'il a du système et des services qui lui sont rendus, des ressources mises réellement en jeu. C'est justement dans cette distanciation que le système maintient un niveau de service que des disfonctionnements locaux auraient tendance à malmener.

Pour simplifier l'étude et procéder de façon incrémentale nous fixons que les clients ont des besoins constants en volume d'information (débit) demandé à chaque requête. Cela signifie que l'on travaille avec des besoins moyens. Il est possible d'introduire de l'aléas dans les simulations, mais l'analyse de ces dernières serait d'autant plus difficile que cette pratique serait généralisée. En revanche, nous introduirons ultérieurement de l'aléas maîtrisé (au niveau du débit d'un client particulier) pour étudier par exemple la résistance du modèle à des comportements marginaux, jouant ainsi le rôle de perturbateur. Dans une application de ces techniques à des fins de sécurité, l'introduction de comportements perturbateurs devrait permettre de détecter les faiblesses du système à protéger.

UN PROTOCOLE «AGENT» POUR LA RECHERCHE DE SERVEURS

Nos clients vont donc créer systématiquement (à chaque top de fonctionnement) des requêtes dont le volume de la demande (débit) correspond à leur besoin.
Chaque requête est un agent médiateur entre un client et des serveurs. Elle possède une réelle «autonomie» dans la mesure où:

- sa mission est de trouver un serveur disponible, en capacité de satisfaire sa demande et le plus proche possible de son client commanditaire;
- elle est capable de se déplacer;
- elle dispose d'un appareillage sensoriel[6] lui permettant de voir les serveurs qui évoluent dans le domaine où elle se trouve[7].

Le contexte du système est donc constitué de clients, d'agents «requêtes» et de serveurs situés dans un espace à deux dimensions, alloué au domaine courant.
Seuls les serveurs et les requêtes changent de position, au cours du temps, dans cet espace.

---

5 CELA N'EST PAS NOUVEAU, NOUS EN VOULONS POUR PREUVE LE PROTOCOLE IP DE TRANSMISSION D'INFORMATION QUI GARANTIT LE CHEMINEMENT DES PAQUETS D'UNE MACHINE À UNE AUTRE QUELQUE SOIT L'ENDOMMAGEMENT LOCAL DU RÉSEAU ET PAR CONSÉQUENT LE CHEMIN SUIVI. CE PROTOCOLE IP PEUT ÊTRE VU PAR ANALOGIE COMME L'ANCÊTRE DE NOS «AGENTS ÉMISSAIRES».

6 DANS *oRis* UNE FONCTION PERMET À N'IMPORTE QUEL OBJET ACTIF DE RECHERCHER L'OBJET D'UN TYPE DONNÉ, QUI LUI EST LE PLUS PROCHE. LE NOM DE CET OBJET CIBLE SUFFIT À L'OBJET DE DÉPART POUR ÉTABLIR UNE COMMUNICATION DIRECTE (INVOCATION DE MÉTHODE) OU INDIRECTE (DÉPLACEMENT VERS LA CIBLE) AVEC LUI.

7 ON RETROUVE CERTAINES DES PRINCIPALES CARACTÉRISTIQUES (MOBILITÉ ET SYSTÈME SENSORI-MOTEUR BOUCLÉ) QUE LES «COMPARTEMENTALISTES» (COMME VARELA,…) ATTRIBUENT COMME UNE NÉCESSITÉ POUR TOUT SYSTÈME ADAPTATIF.

Lorsqu'un serveur du type recherché passe dans le champ de vision d'une requête, cette dernière se dirige dans sa direction suivant un certain pas de déplacement[8]. À chaque pas d'évolution, la requête réexamine son champ de vision pour suivre le serveur le plus proche[9]; ceci lui permet de s'adapter à un contexte changeant (déplacement ou disparition du serveur visé, ou encore apparition d'un nouveau serveur). En utilisant son dispositif sensoriel, la réexamination systématique du contexte local par la requête n'engendre ni coûts de calcul importants, ni grande capacité de stockage pour mémoriser l'état du contexte (accointances, disponibilité des ressources physiques et logicielles,…). En revanche ce dispositif permet une très grande réactivité aux changements dans le contexte.

Dans nos simulations, l'appareillage sensoriel des requêtes, construit au-dessus de celui d'*oRis*, est piloté par l'objet *Moniteur* qui actualise en permanence des informations sur les objets qui évoluent dans le domaine qu'il gère. Dans les systèmes répartis ouverts (SRO) de type *CORBA*, nous avons travaillé depuis longtemps sur la notion de gestionnaire d'objets (GO) dont le rôle est justement celui de l'objet *Moniteur*. Nous en avons fait plusieurs versions qui montrent la pertinence de cette approche, y compris dans un environnement standardisé comme *CORBA*. De tels GO forment le dispositif de base à tout appareil sensoriel dans les SRO. Des sophistications sont possibles et ont même déjà été entreprises. C'est le cas des courtiers ("trader" dans *CORBA*), ou encore des mécanismes de structuration de l'espace de recherche.

Lorsque la requête arrive sur la machine du serveur visé, elle interroge le serveur pour savoir s'il est capable de lui fournir, en quantité suffisante, l'information demandée (les besoins du client). Si c'est le cas, la requête ouvre une session entre son client et le serveur (en bleu sur la figure *Fig-IV-B2-1*). Si le serveur n'est pas disponible (réserve insuffisante…) alors la requête continue son exploration pour trouver un autre serveur (en rouge sur la figure *Fig-IV-B2-1*). Lorsque la requête rencontre cet autre serveur, un lien d'interaction résiduelle est mis en place dans les données internes de ces deux serveurs. Cette notion nous sert à mesurer globalement (comme un observateur extérieur au système) l'efficacité de la ré-

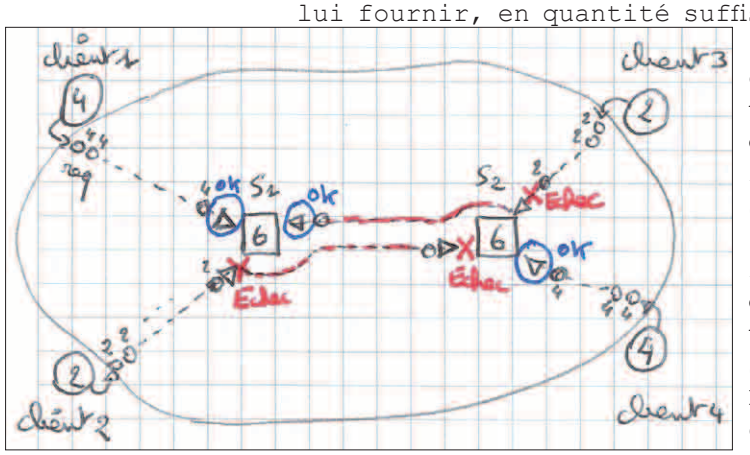


Fig-IV-B2-1

8 Dans nos simulations le pas de déplacement est soit constant, soit variable en fonction de l'objet cible et de l'objet qui se déplace; différentes solutions sont donc envisageables dans la mesure où elles correspondent aux hypothèses prises pour simuler le réseau. Une brève étude de ce phénomène sera présentée dans la partie IV-E2.

9 Nous nous appuyons ici sur les résultats obtenus dans l'étude théorique de $C_N S_K$ (vue au chapitre III), à savoir les notions de *K-partition* «utile» et de multi-points MM_{K-P} «utile», qui sont conformes à cette idée «du serveur le plus proche» pour découper le système en régions «utiles», c.-à-d. minimisant les distances entre les clients et le serveur de la région courante.

organisation du système, dont l'un des objectif est de converger dans un état du système qui minimise des critères, comme par exemple le nombre des interactions résiduelles, la distance (ou temps) moyenne parcourue par requête,…

Il existe des variantes à ce protocole. À chaque fois que l'agent requête subit un échec avec un serveur donné, soit il marque ce serveur dans une liste afin de ne pas le choisir à nouveau, soit il ne marque que le dernier rencontré.
Ce dernier cas permet de mieux réagir à une indisponibilité conjoncturelle du serveur le plus proche.

Comme évoqué plus haut, nous devons conduire des études sur les modalités de déplacement des requêtes et des serveurs.
Les déplacements d'objets autonomes nous confrontent à des échelles temporelles potentiellement différentes nécessitant des synchronisations pour atteindre nos objectifs globaux de stabilité.

Nous connaissons parfaitement ce genre de situation dans les systèmes d'exploitation des ordinateurs. En effet les processeurs en mémoire centrale ne fonctionnent pas du tout dans la même échelle temporelle que les périphériques de stockage, comme les disques durs externes. Cela contraint le système à mettre en œuvre des mécanismes de compensation (ordonnancement, buffer cache,…).
L'arrivée trop massive de requêtes sur les serveurs peut conduire à une réorganisation trop réactive, incapable de recherche d'équilibre et de stabilité. C'est ce que nous verrons dans nos simulations.

IV-B3) L'AUTO-AJUSTEMENT DE LA PRODUCTION

L'auto-ajustement de la production est un mécanisme qui fait partie du comportement de chaque serveur dans le domaine. L'objectif pour chaque serveur, est d'ajuster périodiquement sa production aux demandes (requêtes) qui lui sont faites, en tenant compte des capacités du site sur lequel il se trouve et de la dynamique dans le temps de ces demandes.

Le calcul d'auto-ajustement est fait par les serveurs eux-mêmes entre deux périodes d'observation (cf. la figure *Fig-IV-B3-2*). Une période d'observation (PO) est un ensemble de tops d'horloge consécutifs, défini dynamiquement par chaque serveur.
Le dernier top (hachuré sur la figure *Fig-IV-B3-2*) permet au serveur d'adapter son comportement en fonction de ce qu'il a observé du système pendant les tops de la période courante.
Les autres tops lui permettent de répondre aux besoins des clients.

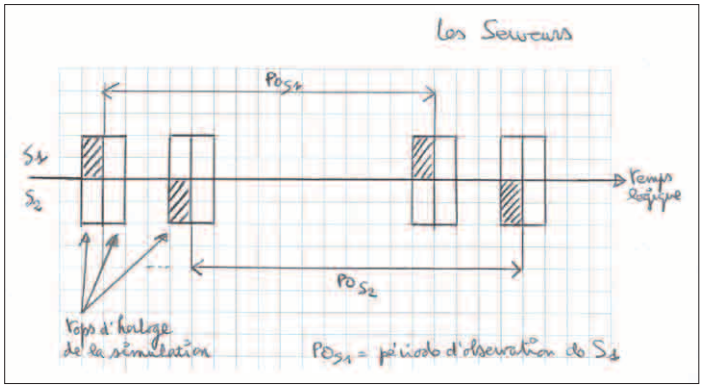


Fig-IV-B3-2

Il s'agit de calculer une valeur d'ajustement (*Ajustement*) de la production pour la période d'observation suivante, afin de faire tendre sa production vers la demande. Cette valeur peut être négative (sur-production) ou positive (sous-production).

Elle est calculée par la formule suivante:

```
Ajustement=(VolT_dem-(VolT_prod+Reserve))/PERIODE_OBS   (1)
```

où:

- *VolT_dem* représente, pour la période d'observation qui vient de se terminer, le cumul des demandes des clients sur ce serveur;
- *VolT_prod* représente, pour cette même période, le cumul de ce que ce serveur a produit;
- *Reserve* représente la réserve de production du serveur;
- *PERIODE_OBS* représente le nombre de pas d'évolution de la période d'observation.

Dans la mesure où les serveurs peuvent se déplacer, ils pourront aller sur des machines aux capacités propres[10]. Cet auto-ajustement garantit que les serveurs sont en adéquation avec les demandes qu'ils reçoivent et les capacités des sites sur lesquels ils fonctionnent. Cela signifie que, sauf pour des serveurs qui gèrent des files d'attente, la limitation de traitement n'est pas liée au serveur lui-même[11], mais à la capacité de la machine sous-jacente. En revanche cela n'écarte pas une éventuelle mauvaise utilisation des machines dans l'ensemble du domaine.

Même si une machine peut supporter un certain nombre de demandes, il n'est pas bon qu'elle fonctionne à la limite de ses capacités, surtout si d'autres machines dans le domaine sont largement sous-utilisées. C'est pourquoi, nous avons rajouté à ce mécanisme une limite maximale de production, non pas attachée à une machine, mais à chaque serveur. Cette valeur dépend du nombre de serveurs présents dans le domaine et de la valeur maximale de production du domaine entier.

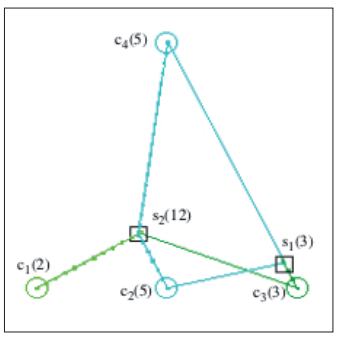


Fig-IV-B3-3

Sur la figure *Fig-IV-B3-3* le domaine est constitué de quatre clients (c1, c2, c3, c4) et de deux serveurs (s1, s2). Les clients ont des besoins constants (en unité de production) et les serveurs ont une valeur initiale de production quelconque. Le domaine dispose d'une capacité globale de traitement de quarante unités par «pas d'évolution».
La valeur maximale autorisée par serveur est de 40/2, soit vingt p r serveur. Cette valeur est indépendante du nombre de machines sur le domaine. Dès qu'un serveur (logiciel) est créé ou disparaît, cette information est transmise au «Manager»

10 Comme nous l'avons annoncé plus haut, dans cette version toutes les machines ont la même capacité; ajouter des capacités différentes ne pose pas de problème particulier.

11 Nous avons fait l'hypothèse de travailler avec des serveurs logiciels sans état vis-à-vis du client. Dans ce cas nous n'avons pas de problème d'accès concurrent aux données du serveur et le traitement peut être paralélllisé en utilisant des fils d'exécutions ("thread"). ceci n'interdit pas l'étude de serveurs à états, mais nécessiterait de reconsidérer les mécanismes utilisés dans nos simulations.

du domaine (qui joue le rôle de gestionnaire d'objet) qui recalcule cette limite maximale de production et l'envoie à chaque serveur du domaine. Nous reviendrons sur le rôle de ces composants dans la partie IV-D («Simulations : le code»).

L'effet de ce seuil de production, dont dispose chaque serveur, est de mieux répartir les serveurs sur les différentes machines disponibles dans le domaine. Ce dispositif peut paraître en contradiction avec l'approche autonome et décentralisée du dispositif. Il faut le condidérer comme une centralisation/organisation propre à l'environnement qui permet une amélioration du dispositif, sans que cela soit indispensable. L'idée est de conserver une approche principalement autonome et décentralisée, mais pas exclusivement. Le coût de cette répartition de la charge des nœuds rend l'auto-organisation plus intéressante en terme qualitatif.

Le processus d'apprentissage décentralisé permet aux serveurs de prévoir, en fin de période d'observation, les quantités nécessaires pour la période suivante. On voit cet ajustement en vert sur la figure *Fig-IV-B3-4*, où la pente du segment correspond à la quantité de réserves nécessaire pour la prochaine période. On voit également le rôle très important de la longueur de la période d'observation (PO) dans une bonne adaptation aux changements éventuels des demandes des clients.

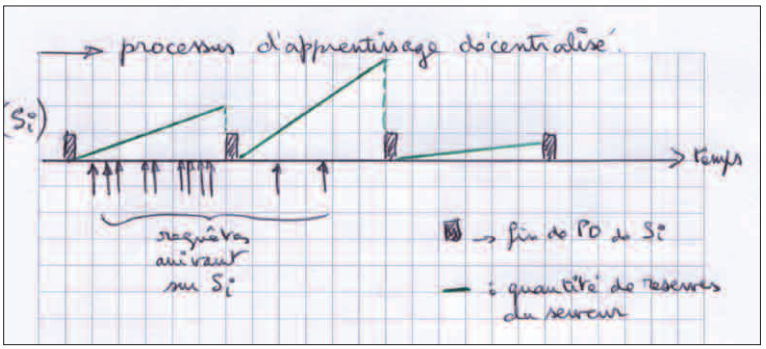


Fig-IV-B3-4

L'idée est que la fonction d'ajustement, propre à chaque serveur, converge le plus rapidement possible vers 0 (le segment vert devenant horizontal).

En planification on cherche à exploiter l'historique de ce qui s'est passé pour calculer la politique optimale permettant de prendre les bonnes décisions face aux choix qui se présentent ; c'est ce que l'on appelle l'exploitation. Les résultats théoriques montrent que plus l'historique est grand (vers le passé) et plus la politique optimale sera fine. En revanche plus il est grand et plus les temps de calculs associés à cette politique seront conséquents. Donc, en général, on raccourcit l'historique pour des questions d'efficacité et de calculabilité.
Il s'avère, d'après nos simulations (dans la partie IV-E), que l'historique aurait une valeur optimale, ni trop courte, ni trop longue. Contrairement à l'exploitation en planification, si l'historique est trop grand il dégrade la nature de la solution obtenue par les heuristiques résolutives à C_nS_k.
Cela provient probablement du fait que ces heuristiques font aussi de l'apprentissage (exploration). Nous allons détailler ci-après cette fonction d'apprentissage et ses propriétés de convergence sous conditions.

L'historique doit donc avoir une borne inférieure en deçà de laquelle le système devient réactif.
La concordance entre ces deux exigences (exploitation/ exploration) revient à minimiser la durée de l'historique pour minimiser les erreurs d'appréciations induites (exploration), et à maximiser cette même longueur pour affiner ces appréciations (exploitation). Devant cette contradiction apparente nous voyons apparaître la notion de fenêtre d'historique, c.-à-d. la longueur optimale associée à une marge d'erreur.

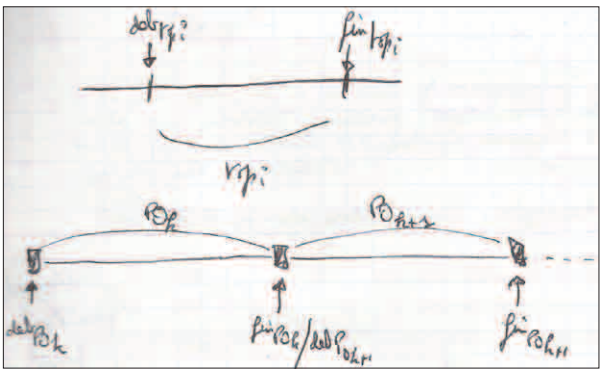


Fig-IV-B3-5

Nous allons revenir sur les propriétés du mécanisme d'ajustement pour préciser son fonctionnement.

À la fin de chaque pas de simulation (top_i) chaque serveur a produit/réservé une certaine quantité de ressources (*Production*) ; il met à jour sa réserve de production (*Reserve*) ainsi que le volume total cumulé des ressources (*VolT_prod*) qu'il a réservées/ produites depuis le début de la période d'observation (PO_k sur la figure *Fig-IV-B3-5*) courante :

Reserve=Reserve+Production, *(2)*
VolT_prod=VolT_prod+Production. *(3)*

En considérant :

- la fonction *Longueur(PO_k)* qui calcule le nombre de pas de simulations (top_i) de la période d'observation PO_k ;
- *VolT_dem* comme étant le volume total cumulé des demandes faites au serveur courant depuis le début de la période d'observation (PO_k) ;
- *Ajustement* l'ajustement de la production/réservation de ressources calculée en fin de PO_k et appliquée en début de période d'observation suivante (PO_{k+1}).

Nous avons, à la fin de chaque période d'observation (PO_k), les grandeurs suivantes qui vont être mises à jour :

Ajustement=(VolT_dem-(VolT_prod+Reserve))/Longueur(PO_k), *(4)*
Production=Production+Ajustement. *(5)*

Nous avons vu que les agents requête sont des médiateurs entre les clients et les serveurs, puisqu'ils sont en charge de trouver les serveurs adéquats et de négocier avec eux pour satisfaire les besoins de leur client.

Dès qu'un serveur adéquat (c.-à-d. disposé à répondre au type de demande du client) a été trouvé par la requête (après une exploration du système que nous détaillerons dans la partie IV-D), cette dernière lui transmet la quantité demandée (*Demande*). Le dit serveur ajoute cette nouvelle demande à son volume cumulée de demandes (*VolT_ dem*). Si sa réserve (*Reserve*) est suffisante, alors cette dernière est mise à jour de la façon suivante :

Reserve=Reserve-Demande.

Dans la mesure où un serveur donné peut être sollicité simultanément par plusieurs requêtes, nous devons tenir compte de cela dans les équations.

Nous obtenons :

$$Reserve(fin_{top_i}) = Reserve(deb_{top_i}) - \sum_{j=1}^{m} Demande(j, top_i) + Production(fin_{top_i}) \quad (6)$$

où 'm' et 'p' correspondent respectivement au nombre de demandes pendant chaque top_i et pendant chaque période d'observation PO_k.

Nous pouvons voir dans *(7)* que la réserve de production, en fin de période d'observation PO_k, correspond à la réserve en début de période plus la production pendant la période et moins l'ensemble des demandes pendant cette même période.

$$Reserve(fin_{PO_k}) = Reserve(deb_{PO_k}) - \left[\sum_{j=1}^{p} Demande(j, PO_k)\right] + \left[Longueur(PO_k) \times Production(PO_k)\right] \quad (7)$$

$$Reserve(fin_{PO_k}) = \sum_{i=1}^{Longueur(PO_k)} Reserve(fin_{top_i}) \quad (8)$$

$$Ajustement(fin_{PO_k}) = \frac{VolTdem(fin_{PO_k}) - [VolTprod(fin_{PO_k}) + Reserve(fin_{PO_k})]}{Longueur(PO_k)} \quad (9)$$

$$Production(PO_{k+1}) = Production(PO_k) + Ajustement(fin_{PO_k}) \quad (10)$$

$$VolTprod(fin_{PO_k}) = \sum_{i=1}^{Longueur(PO_k)} Production(fin_{top_i}) \quad (11)$$

$$VolTdem(fin_{PO_k}) = \sum_{j=1}^{p} Demande(j, PO_k) \quad (12)$$

L'ajustement correspond à la correction de l'erreur entre la production et la demande. Il est inversement proportionnel à la longueur de la période d'observation (*Longueur(*PO_k*)*).

Plus cette valeur est grande et plus l'ajustement est lent, quelque soit l'erreur faite. On dira que l'ajustement ne fonctionne plus par manque de convergence vers 0.

Pour simplifier l'écriture des développements à suivre, nous allons opérer un changement de notation de ces grandeurs :

Ajustement(fin_{PO_k})	= a_{k+1},	// ajustement calculé en fin de PO_k
Reserve(fin_{PO_k})	= R_k,	// réserve en fin de PO_k
Production(PO_{k+1})	= p_{i_k},	// production à chaque top_i de PO_k
VolT_dem(fin_{PO_k})	= D_k,	//ensemble des demandes sur PO_k
VolT_prod(fin_{PO_k})	= P_k,	//ensemble des productions sur PO_k
Longueur(PO_k)	= L_k.	//nombre de top_i dans PO_k

Nous obtenons les équations suivantes:

$R_k=R_{k-1}-D_k+P_k$, avec *(13)*

$P_k=\sum_{i=1}^{Lk} p_{i\_k}$. *(14)*

P_k est la production faite sur l'ensemble de la période d'observation PO_k. Elle intègre l'ajustement (a_k) de l'erreur de production précédente (PO_{k-1}), calculée en fin de PO_{k-1}.
D_k est l'ensemble des demandes faites en fin de PO_k.
R_{k-1} est la quantité de ressources produites et non consommées en fin de PO_{k-1}.
p_{i_k} est la production faite dans le top_i (unité de fonctionnement) de PO_k; on notera $p_{i\_k}=p_k$.

Si la production est constante sur l'ensemble des top_i de PO_k, alors nous obtiendrons:

$P_k=L_k \times p_k$, *(15)*

$a_{k+1}=(D_k-(P_k+R_k))/L_k$, *(16)*

$A_k=L_k \times a_k=D_k-(P_k+R_k)$, *(17)*

$p_{k+1}=p_k+a_{k+1}$. *(18)*

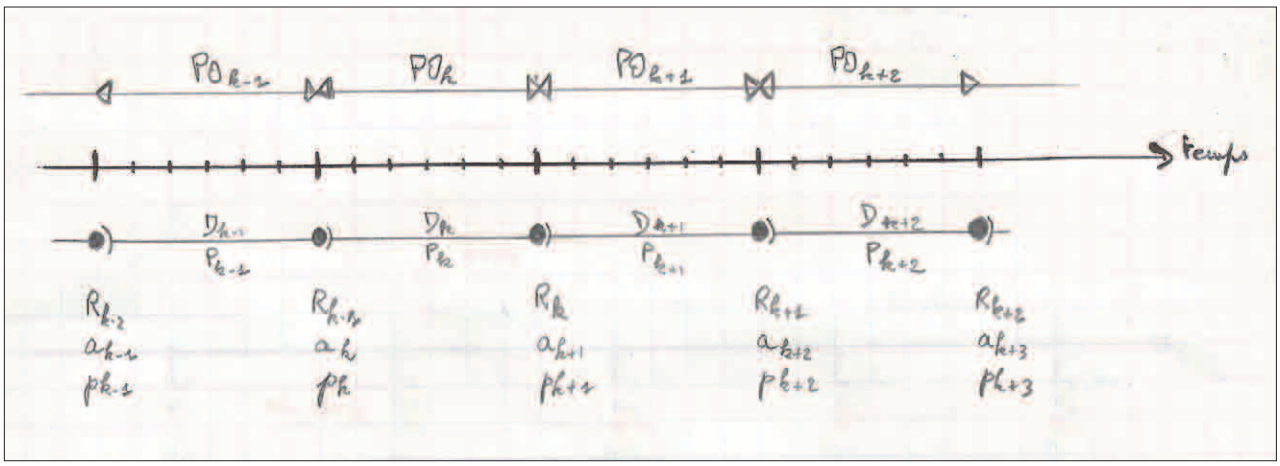


Fig-IV-B3-6

Sans détailler les formules regardons comment fonctionne la prise en compte de la réserve de production dans le calcul de l'ajustement:

1. $D_k=P_1$, $\forall k \Rightarrow$ la demande cumulée en fin de PO_k est constante.

$R_1=R_0-D_1+P_1=-P_1+P_1=0$,
$a_2=[D_1-P_1-R_1]/L1=P_1-P_1/L_1=0$,
or $p_2=p_1+a_2=p_1$,
si L_k (=L) est constante $\forall k \Rightarrow L_2=L_1=L$ et $P_2=L \times p_2=L \times p_1=P_1$,
$R_2=R_1-D_2+P_2=-P_1+P_2=0$,
$a_3=(D_2-(P_2+R_2))/L_2=(P_1-P_1)/L_2=0$.

Par récurrence on montre dans ce cas que
$\forall n$, $P_n=P_1$ et $a_n=R_n=0$.

Dans chaque période d'observation PO_k, la somme des demandes est constante et elle correspond à la somme des ressources produites dans PO_k.
Il n'y a ni réserve, ni ajustement (cf. les traits horizontaux en noir sur la figure *Fig-IV-B3-7*).

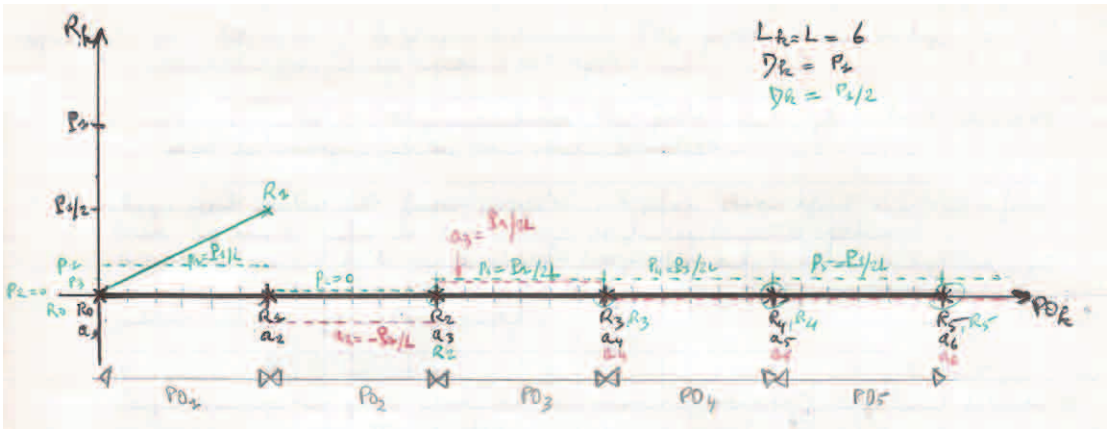


Fig-IV-B3-7

2. $D_k=P_1/2$, $\forall k \Rightarrow D_1=P_1/2$

$R_1=R_0-D_1+P_1=-P_1/2+P_1=P_1/2$,
$a_2=[D_1-P_1-R_1]/L1=-P_1/L_1$,
$p_2=p_1+a_2=p_1-(P_1/L_1)$ avec $p_1=P_1/L_1 \Rightarrow p_2=P_2=0$,
$R_2=R_1-D_2+P_2=P_1/2-P_1/2=0$,
$a_3=(D_2-(P_2+R_2))/L_2=P_1/2L_2$, si L_k (=L) est constante $\forall k \Rightarrow a_3=P_1/2L$,
$p_3=P_1/2L$,
$R_3=-P_1/2+P_1/2=0$,
$a_4=(P1/2-P1/2)/L_3=0$,
$p_4=P_1/2L$.

Par récurrence on montre dans ce cas que
$\forall n \geq 3$ $R_n=a_n=0$ et $p_n=P_1/2L$.
Cette situation est représentée en vert sur la figure *Fig-IV-B3-7*. Le mécanisme d'ajustement corrige très vite l'erreur puisque pour $\forall n \geq 3$ $R_n=a_n=0$.
En ligne pointillée de couleur verte, sur la figure *Fig-IV-B3-7*, nous avons l'évolution de la production sur chaque PO_k. Dans la mesure où la demande (constante) vaut la moitié de la production initiale P_1 (PO_1), l'ajustement en ligne pointillée rouge, sur la figure *Fig-IV-B3-7*, corrige la production pour diminuer la réserve.
Avec PO_3 le processus d'ajustement a convergé ($a_n=R_n=0$) et $p_n=P_1/2L$.

3. Quid si R_k ne tient pas compte de R_{k-1}?

Dans pareil cas nous obtenons par récurrence:
$\forall n \geq 1$, $P_{2n}=-P_1/2$, $a_{2n+1}=P_1/L$, $p_{2n+1}=P_1/L$, $P_{2n+1}=P_1$,
$P_{2n+1}=P_1/2$, $a_{2(n+1)}=-P_1/L$, $p_{2(n+1)}=0$, $P_{2(n+1)}=0$.

On voit, sur la figure *Fig-IV-B3-8*, qu'à la fin de PO_1 on a produit $P_1/2$ quantité de ressources en trop ($\Rightarrow R_1$). Comme on n'intègre pas cet excédent dans le calcul de R_2, la correction $a_2=-P_1/L$ est trop forte… le système oscille sans jamais atteindre un état stable.

Pour aller vers la généralisation du processus d'ajustement, nous allons reprendre le cas où la demande cumulée est constante à l'intérieur de chaque période PO_k.
Cette fois cette demande dépend, via une fonction 'f', de la production P_k propre à chaque période d'observation.

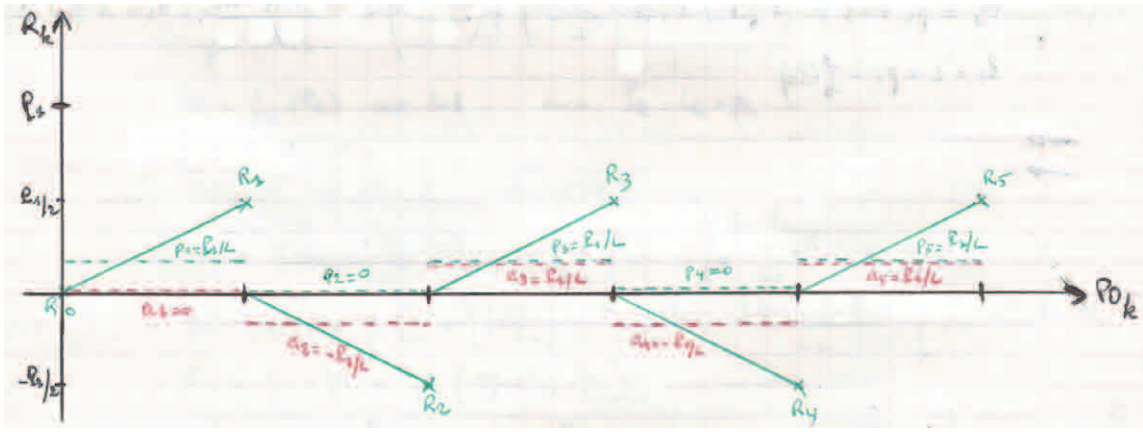


Fig-IV-B3-8

Nous avons $\forall k$ $D_k=D$ (constante), avec $D=f(P_1)$ et $L_k=L$ (constante). Les formules *(13)* et *(16)* se réécrivent de la façon suivante:

$R_k=R_{k-1}-f(P_1)+P_k$, avec *(19)*

$a_{k+1}=(f(P_1)-(P_k+R_k))/L_k$. *(20)*

On obtient:

$R_1=P_1-f(P_1)$,
$a_2=-2R_1/L$,
$p_2=(P_1-2R_1)/L$,
$P_2=2f(P_1)-P_1$,
$R_2=f(P_1)-f(P_2)$,
$a_3=R_1/L$,
$p_3=f(P_1)/L$,
$P_3=f(P_1)$,
$R_3=a_4=0$,
$p_4=f(P_1)/L$.

Par récurrence on obtient:

$\forall n\geq 3$ $R_n=a_n=0$, $p_n=f(P_1)/L$ et $P_n=f(P_1)$.

Mêmes résultats que précédemment, $\forall f(P_1)=D$ (constante) et $\forall k$ $L_k=L$ (constante), le processus d'ajustement converge en trois étapes (PO_k).

Propriétés: Si $L_k\rightarrow 1$ alors la période d'observation PO_k possède très peu de top_i ($\rightarrow 1$). Dans ce cas la convergence de l'ajustement est très rapide, mais l'opération devra être renouvelée souvent si les variations des demandes s'étalent dans le temps. Le processus est efficace, mais revient souvent.
Si $L_k\rightarrow\infty$, à l'inverse PO_k possède un très grand nombre de top_i. La convergence peut ne jamais aboutir.

Chaque agent serveur utilise ces propriétés. A priori L_k pourrait être choisie assez courte pour accélérer la convergence vers un état stable. Nous verrons plus loin que la coopération, nécessaire entre les serveurs pour faire converger l'ensemble du système C_nS_k vers un état global stable, impose des synchronisations entre eux.

Globalement des L_k trop courtes conduisent à des systèmes réactifs, alors que des L_k trop longues donnent des systèmes qui procrastinent.

Il faut noter que la convergence du système vers un état stable n'est pas nécessairement l'état optimal recherché. En effet nous verrons, dans les simulations, que d'autres critères peuvent être pris en compte pour l'optimalité de la gestion des ressources (nombre d'agents requête, distances parcourues,…).

Cas général

Regardons de plus près les formules dans le cas général du processus d'ajustement (D_k et L_k variables d'une PO_i à une autre PO_j).

Remarque : La demande globale D_k, correspondant à la période d'observation PO_k, est calculée en fin de PO_k. Les demandes peuvent fluctuer d'une période à une autre, mais la notion de demande constante sur PO_k ne veut pas dire régulière sur cette période (effet cumulatif en fin de PO_k).

Nous avons : $D_k=f_k(P_k)$, L_k et $P_k=L_k \times p_k$. Soit :

$R_1=P_1-f_1(P_1)$,
$a_2=-2/L1(P_1-f_1(P_1))$,
$p_2=(P_1-2R_1)/L_1$,
$P_2=(L_2/L_1)[2f_1(P_1)-P_1]$,
$R_2=[(L_1-L_2)/L_1]P_1+[(2L_2-L_1)/L_1]f_1(P_1)-f_2(P_2)$.

On retrouve bien ici la convergence sous-jacente en trois étapes. En effet si $D_k=f_k(P_k)=D$ (constante) et $L_k=L$ (constante) $\forall k$, alors $R_2=0$.
En continuant les calculs sans présenter toutes les étapes intermédiaires nous obtenons pour les premières valeurs de a_n (2 à 7) :

$$a_2=(-2/L_1)P_1+(2/L_1)f_1(P_1),$$

$$a_3=[(2L_2-L_1)/L_2L_1]P_1+[(-4L_2+L_1)/L_2L_1]f_1(P_1)+[2/L_2]f_2(P_2),$$

$$a_4=[(2L_3-L_2)(-L_2+L_1)/L_3L_2L_1]P_1+[(2L_3-L_2)(2L_2-L_1)/L_3L_2L_1]f_1(P_1)+[(-4L_3+L_2)/L_3L_2]f_2(P_2)+[2/L_3]f_3(P_3),$$

$$a_5=[(2L_4-L_3)(-L_3+L_2)(-L_2+L_1)/L_4L_3L_2L_1]P_1+[(2L_4-L_3)(-L_3+L_2)(2L_2-L_1)/L_4L_3L_2L_1]f_1(P_1)+[(2L_4-L_3)(2L_3-L_2)/L_4L_3L_2]f_2(P_2)+[(-4L_4+L_3)/L_4L_3]f_3(P_3)+[2/L_4]f_4(P_4),$$

$$a_6=[(2L_5-L_4)(-L_4+L_3)(-L_3+L_2)(-L_2+L_1)/L_5L_4L_3L_2L_1]P_1+[(2L_5-L_4)(-L_4+L_3)(-L_3+L_2)(2L_2-L_1)/L_5L_4L_3L_2L_1]f_1(P_1)+[(2L_5-L_4)(-L_4+L_3)(2L_3-L_2)/L_5L_4L_3L_2]f_2(P_2)+[(2L_5-L_4)(2L_4-L_3)/L_5L_4L_3]f_3(P_3)+[(-4L_5+L_4)/L_5L_4]f_4(P_4)+[2/L_5]f_5(P_5),$$

$$a_7=[(2L_6-L_5)(-L_5+L_4)(-L_4+L_3)(-L_3+L_2)(-L_2+L_1)/L_6L_5L_4L_3L_2L_1]P_1+[(2L_6-L_5)(-L_5+L_4)(-L_4+L_3)(-L_3+L_2)(2L_2-L_1)/L_6L_5L_4L_3L_2L_1]f_1(P_1)+[(2L_6-L_5)(-L_5+L_4)(-L_4+L_3)(2L_3-L_2)/L_6L_5L_4L_3L_2]f_2(P_2)+[(2L_6-L_5)(-L_5+L_4)(2L_4-L_3)/L_6L_5L_4L_3]f_3(P_3)+[(2L_6-L_5)(2L_5-L_4)/L_6L_5L_4]f_4(P_4)+[(-4L_6+L_5)/L_6L_5]f_5(P_5)+[2/L_6]f_6(P_6).$$

La formule générale peut s'écrire de la façon suivante:
$a_n=\alpha_{n\_0}\times f_0(P_0)+\alpha_{n\_1}\times f_1(P_1)+\alpha_{n\_2}\times f_2(P_2)+\ldots+\alpha_{n\_(n-1)}\times f_{(n-1)}(P_{(n-1)})$ *(21)*

Elle a, ∀n>2, 'n' termes α_{n_i} avec i∈[0,n-1].

Nous observons, grâce au code couleur, sur ces six premières valeurs, que la formule générale comporte un premier terme ( ), trois derniers termes ( ) et (n-4) termes intermédiaires ( ).

Par récurrence nous obtenons les formules générales de l'ajustement a_n, ∀n>3:

- le premier terme α_{n_0},

$$\alpha_{n\_0} = (2L_{n-1} - L_{n-2}) \times \left[\frac{\prod_{i=2}^{n-2}(-L_i + L_{i-1})}{\prod_{j=1}^{n-1}(L_j)}\right] \qquad (22)$$

- les trois derniers termes $\alpha_{n\_(n-1)}$, $\alpha_{n\_(n-2)}$ et $\alpha_{n\_(n-3)}$,

$$\alpha_{n\_(n-1)} = \frac{2}{L_{n-1}} \qquad \text{pour tout } n \geq 2 \qquad (23)$$

$$\alpha_{n\_(n-2)} = \frac{-4L_{n-1} + L_{n-2}}{L_{n-1} \times L_{n-2}} \qquad \text{pour tout } n \geq 3 \qquad (24)$$

$$\alpha_{n\_(n-3)} = \frac{(2L_{n-1} - L_{n-2}) \times (2L_{n-2} - L_{n-3})}{L_{n-1} \times L_{n-2} \times L_{n-3}} \qquad \text{pour tout } n \geq 4 \qquad (25)$$

- les (n-4) autres termes (∀n>4) écrits sous la forme d'une somme 'S'.

$$S = \sum_{k=0}^{n-5}(\alpha_{n\_(k+1)}) \times f_{k+1}(P_{k+1}) \qquad \text{avec} \qquad (26)$$

$$\alpha_{n\_(k+1)} = (2L_{n-1} - L_{n-2}) \times \left[\frac{\prod_{i=1}^{(n-4)-k}(-L_{i+k+2} + L_{i+k+1})}{\prod_{j=k+1}^{n-1}(L_j)}\right] \times (2L_{k+2} - L_{k+1}) \qquad (27)$$

Cette formule générale (*(21)* à *(27)*) confirme que seuls les trois derniers termes (*(23)* à *(25)*), correspondant aux trois périodes d'observation PO_k les plus récentes, fournissent un ajustement $a_n \neq 0$ si les périodes d'observation ont une longueur constante ($L_k=L$), avec des demandes $f_k(P_k)$ qui ne le sont pas. Dans ce cas nous obtenons:

$a_n=1/L\times[f_{n-3}(P_{n-3})-3f_{n-2}(P_{n-2})+2f_{n-1}(P_{n-1})+]$ avec *(28)*

$R_n=f_{n-1}(P_{n-1})-f_n(P_n)$. *(29)*

L'état d'un agent serveur est considéré comme stable si $a_n=R_n=0$, ∀n. Dans ce cas il ne possède pas de réserve ($R_n=0$) et l'ajustement est mis en pause.
Ces formules générales confirment que ces conditions sont obtenues si la longueur (L_k), en nombre de top_i, des périodes d'observation (PO_k) est constante et si la demande des clients l'est aussi.

La conséquence de ces résultats est que si ces deux conditions ne sont pas remplies, le processus d'ajustement ne peut pas converger vers un état stable, aussi bien au niveau de chaque agent serveur, que de l'ensemble du système (tous les agents serveur).

Condition-1: Pour arriver à cet état stable (ce que nous validerons avec les simulations) il faut conduire les serveurs à rechercher une longueur L_k des périodes PO_k, qui leur soit le «plus favorable» possible et faire en sorte qu'une fois trouvée, elle demeure constante.

Condition-2: Pour obtenir une demande constante sur les serveurs, ce qui n'est pas en contradiction avec une demande variable des agents client, les heuristiques (cf. les simulations) dupliquent et déplacent les agents serveur afin d'arriver à des demandes constantes sur les serveurs, ou une majorité d'entre eux.

Cette deuxième condition nous rappelle que notre système C_nS_k n'est pas composé d'un seul agent serveur, mais d'une population qui évolue (apparition/disparition).
Nous verrons par la suite, que des liens de coopération implicite (via les agents requête) vont apparaître entre les serveurs lors du processus d'ajustement.
Nous allons étudier en détails ces liens en introduisant la notion d'échec des demandes d'un agent requête vers un agent serveur.

Les équations générales de a_n/R_n nous montrent également que l'ajustement ne doit pas se faire sur un historique (en nombre de PO_k) trop long. En réalité les trois dernières PO_k, dans lesquelles les deux conditions sont réunies, suffiront à stabiliser le système. Nous verrons avec les simulations que le système complet ne se stabilise pas d'un seul coup. Toutes les interactions entre les différents agents du système étant parfaitement décentralisées, la stabilité se construit localement et s'étend progressivement à la totalité.

Nous allons voir par la suite que la stabilité d'un agent serveur fait disparaître des propagations d'échecs, qui ont tendance à retarder la stabilité globale en allant jusqu'à introduire des cycles stabilité/instabilité. En devenant stable un agent serveur diminue le temps de convergence du système complet.

Dans la mesure où la première condition (L_k=L) incite les agents serveur à fixer une certaine longueur 'L' de leur PO_k, et que la deuxième condition les incite à coopérer entre eux, les agents serveur vont devoir se synchroniser entre eux autour de leur valeur respective L_k.
Une façon de régler ce problème va être de mettre en place des protocoles d'échange de L_k entre serveurs afin de converger vers des longueurs communes. C'est ce que nous verrons dans les simulations.

Tout l'enjeu des heuristiques résolutives est de conduire le système C_nS_k à satisfaire ces deux conditions.

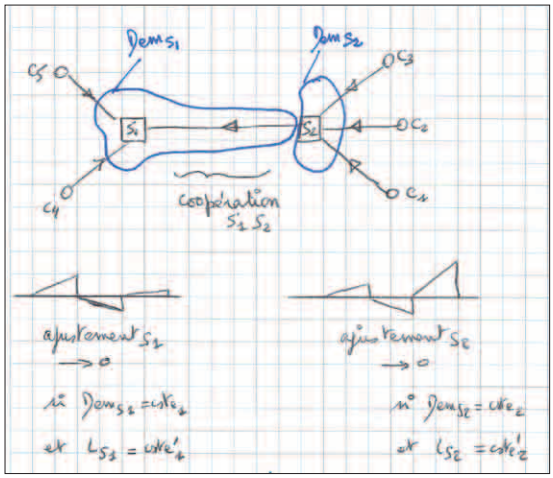


Fig-IV-B4-9

La figure *Fig-IV-B4-9* nous montre des interactions entre les serveurs S_1 et S_2 consécutivement à la mise en place des ajustements a_1 et a_2 par chaque serveur.
Ces interactions (flèche reliant les deux serveurs) se matérialisent par des requêtes en échec sur un serveur ne disposant pas de suffisamment de ressources pour satisfaire la demande.
Ces échecs disparaissent lorsque l'ajustement a terminé son processus de recherche d'un état stable.

L'idée de la restructuration est de minimiser ces interactions entre serveurs, via un processus de coopération. L'objectif est de rendre constantes, autant que faire se peut, les demandes locales à chaque serveur, afin d'accélérer la convergence du processus d'ajustement vers un état stable recherché.

RESTRUCTURATION

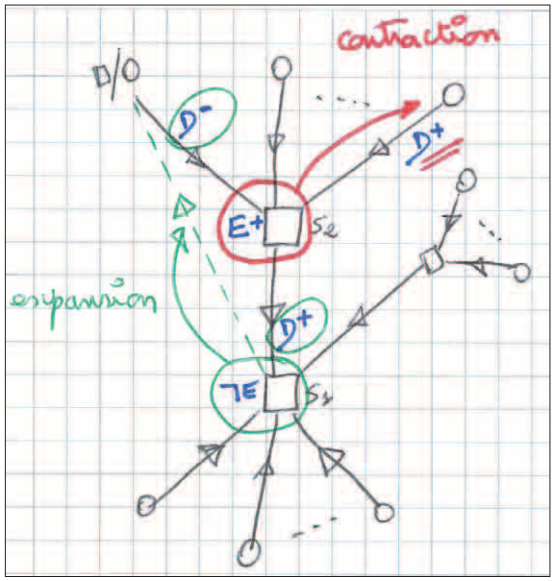


Fig-IV-B4-10

Nous allons voir au travers des simulations, de nombreuses variations de l'algorithme de restructuration de base que nous allons présenter maintenant.
L'algorithme fonctionne sur un principe de contraction/expansion des serveurs vis-à-vis des clients.
Sur la figure *Fig-IV-B4-10* nous avons deux serveurs S_1 et S_2 reliés à des clients respectifs, mais aussi entre eux.
S_2 est en échec (E+) car la somme des demandes (D-,D+,…) qu'il reçoit dépasse sa production en cours. Ce n'est pas le cas pour S_1 (¬E) qui reçoit des demandes consécutivement aux échecs de S_2 (appelées requêtes résiduelles), en plus de ses propres demandes en cours.

S_2 se contracte (en rouge sur la figure *Fig-IV-B4-10*) vers le ou les clients qui le sollicitent le plus, alors que S_1 étend son influence en se déplaçant (en vert sur la figure *Fig-IV-B4-10*) vers les nouvelles demandes. Le serveur S_1 cherche à capturer en direct, sans passer par S_2, les requêtes résiduelles, qui provoquent des échecs sur S_2.

Nous détaillerons cet algorithme avec les simulations et tout particulièrement les critères permettant de lancer des phases de contraction/expansion. Nous pouvons déjà dire qu'il existe un seuil de contraction (% d'échecs) et un seuil d'expansion (% de requêtes résiduelles) au-delà desquels le processus se met en route.
On voit un exemple simple de restructuration, sur la figure *Fig-IV-B4-11*. Le serveur S_1 (E+) produit dix unités de ressources, alors que les demandes qui lui arrivent sont de douze (4+8) unités, d'où des requêtes résiduelles de deux unités vers S_2. Ce dernier (¬E) produit dix unités, alors que ses demandes sont de huit (4+4) unités.
L'algorithme fait se déplacer S_1 vers sa plus grande demande (D+). S_2, quant à lui, se dirige vers le client à l'origine des requêtes résiduelles.
Un équilibre s'installe dès que S_1 arrive sur le cercle centré sur le client à l'origine des requêtes résiduelles et passant par S_2.
Dans cet état le client incriminé peut envoyer ses demandes pour moitié (2) vers S_1 et pour moitié (2) vers S_2.

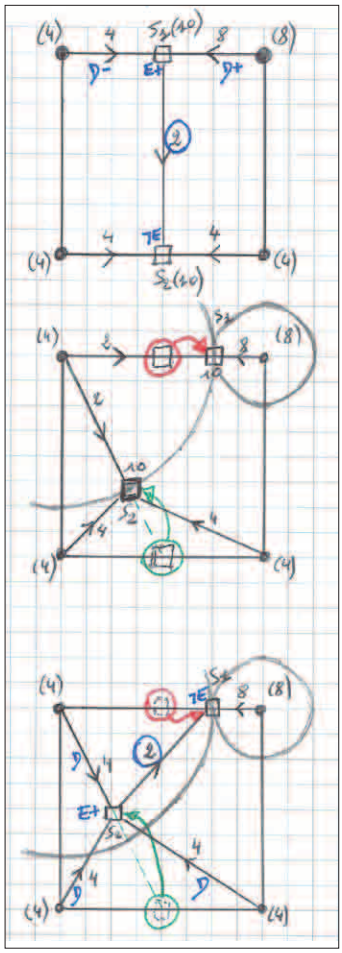


Fig-IV-B4-11

Sur le troisième graphique, de la figure *Fig-IV-B4-11*, nous voyons ce qui se passerait si S_2 continuait d'aller vers le client en question. Nous aurions un dépassement de capacité (12) pour S_2, entraînant à nouveau des échecs, mais cette fois vers S_1. Un phénomène cyclique pourrait alors s'installer.

Requêtes résiduelles, échecs sortants (E↑) et entrants (E↓)

Sur la figure *Fig-IV-B4-12*, nous partons avec une configuration dans laquelle le serveur S_1, au cours de sa première période d'observation PO_{1_1}, de longueur 5 (L_{1_1}=5), ne produit rien (t=0, p=0, R_{1_1}=a_{1_2}=0) alors que la demande cumulée qui lui arrive sur PO_{1_1} est de 20 (D=20). Nous noterons E(+20)↑ l'ensemble des échecs subits par S_1 pendant cette période. La valeur «+20» signifie que le volume des demandes non satisfaites pendant cette première période est de vingt unités de ressources. La flèche '↑' indique que les échecs sont sortants, c.-à-d. que les requêtes résiduelles correspondantes cherchent un serveur disponible.

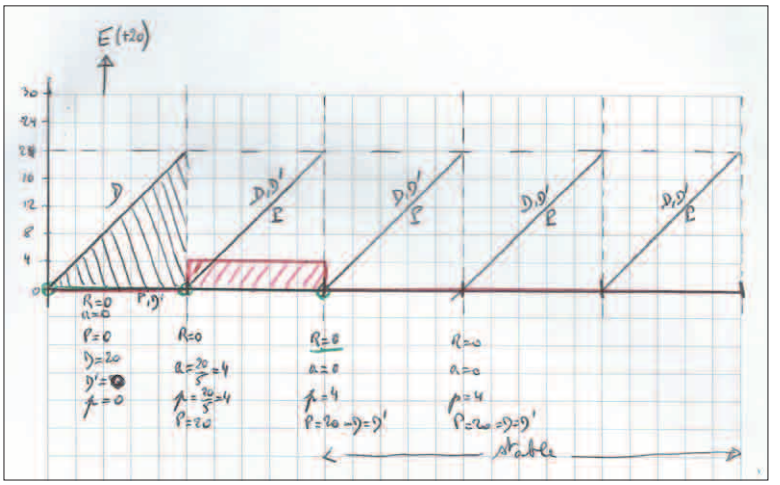


Fig-IV-B4-12

Le mécanisme d'ajustement fonctionne bien puisque pour la deuxième période

PO_{1_2}, l'ajustement a_{1_2} (rectangle rouge hachuré, sur la figure *Fig-IV-B4-12*) vaut 20/5=4. En début de troisième période PO_{1_3} la production de S_1 est stabilisé ($R_{1\_2}=a_{1\_3}=0$). À partir de là, la quantité de demandes effectives D' est identique à celle a priori 'D' et à la production 'P'.

Cette première phase s'est ajustée à la demande en générant des échecs (requêtes résiduelles) lors de la première période d'observation PO_{1_1}.
Si des échecs sont produits cela signifie qu'ils vont arriver sur d'autres serveurs pendant leur fonctionnement. Nous aurons ainsi des requêtes résiduelles qui vont provoquer des échecs sortants (E↑) du serveur de départ et probablement des échecs entrants (E↓) vers le nouveau serveur cible.

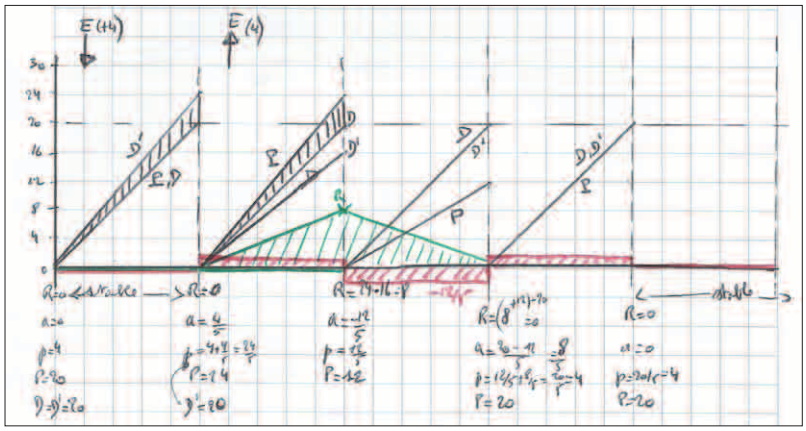


Fig-IV-B4-13

Sur la figure *Fig-IV-B4-13* nous voyons, dans cette deuxième configuration, le phénomène de propagation de requêtes résiduelles (échecs) et sa résorption par le mécanisme d'ajustement. Nous avons une requête résiduelle entrante (E(+4)↓) dans la première période d'observation PO_{2_1}. Cela provoque une demande effective (D'=24) supérieure à la demande a priori (D=P=20).

En fin de première période PO_{2_1} l'ajustement est calculé sur D-D=4, et vaut $a_{2\_2}=4/5$. La production à chaque top de PO_{2_2} sera $p_{2\_2}=4+(4/5)=24/5$.

Suivant l'ordre d'arrivée des demandes pendant cette deuxième période PO_{2_2}, il se peut qu'une demande arrive à un moment où les quantités produites soient insuffisantes. Nous ferons une étude de ce phénomène (ordre des arrivées des demandes) lors des expérimentations avec les simulations.
Dans ce cas la requête subit un échec du serveur. Nous observons une requête résiduelle sortante (E(-4)↑) pendant PO_{2_2}.

Dans ce cas particulier le serveur aura trop produit (+4) pendant PO_{2_2}, générant une réserve $R_{2\_2}=4$ (en vert hachuré sur la figure *Fig-IV-B4-13*) en fin de période. Comme le montre cette figure, le processus d'ajustement met deux périodes supplémentaires (PO_{2_3} et PO_{2_4}) pour atteindre à nouveau un état stable.

Nous aurions pu remplacer le volume des échecs par toute autre valeur, le scénario aurait été le même.

Comme nous le verrons plus loin, ce phénomène de création de requêtes résiduelles est aléatoire dans la mesure où les requêtes sont autonomes entre elles et le déplacement des serveurs rend imprévisible l'ordre d'arrivée et donc de traitement de ces dernières dans les simulations. On retrouve cette propriété dans un système décentralisé réel.

La troisième configuration nous montre d'autres cas de figure intéressants. En effet on voit qu'au cours de la première période d'observation PO_{3_1}, si la quantité d'échecs entrants (E↓) est identique à celle d'échecs sortants (E↑), alors la stabilité du serveur n'est pas impactée: la demande globale 'D' reste identique à la production 'P' et l'ajustement n'est pas enclenché ($R_{3\_1}=a_{3\_2}=0$). Le serveur reste stable tant que les demandes restent synchronisées avec la production.
Ce qui relève du hasard.

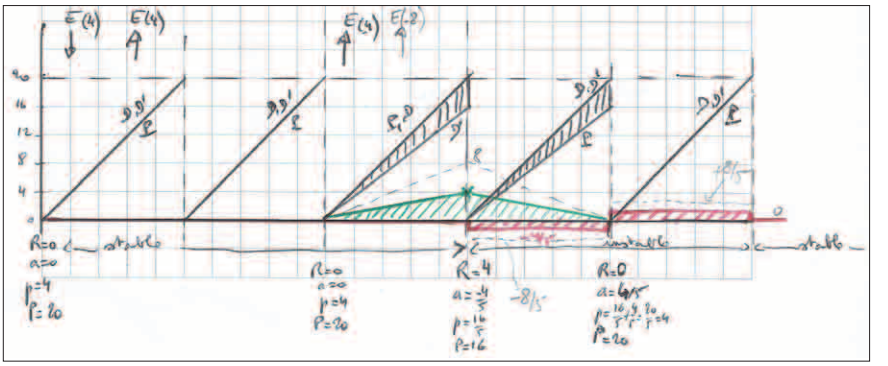


Fig-IV-B4-14

Pendant la troisième période PO_{3_3} deux séries d'échecs sortants (E(-4)↑) et (E(-8)↑) se produisent. La demande globale effective (D') redescend à 8 (20-4-8). Un reste se constitue (en vert) qui sera amorti en fin de période suivante grâce à l'ajustement.

Trois périodes plus tard (PO_{3_6}), s'il n'y a plus d'échec, le serveur sera à nouveau stable. Le phénomène est identique quelque soit le nombre d'échecs sortants (E↑) dans une période d'observation.
La réserve augmentera d'autant qu'il y a des échecs sortants (E↑) en nombre, et l'ajustement oscillera toujours de la même façon mais avec une amplitude plus grande.

Dans ce dernier exemple (sur la figure *Fig-IV-B4-15*) nous abordons le phénomène de neutralisation des échecs entrants (E↓) par les échecs sortants (E↑).

L'arrivé en première période PO_{4_1} d'un échec entrant (E(+4)↓) déclenche pour la période PO_{4_2} suivante le début du cycle d'ajustement.
Lors de la deuxième période PO_{4_2} un nouvel échec entrant (E(+4)↓) arrive.
Or il se trouve que pendant cette même période un échec sortant (E(-4)↑) équivalent se produit.

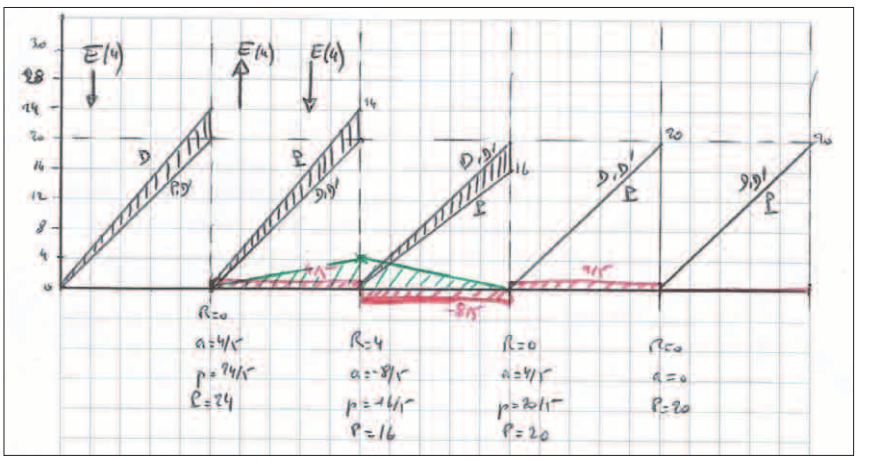


Fig-IV-B4-15

Le résultat est que l'échec entrant (E↓) a été neutralisé par l'échec sortant (E↑), cf. le cas précédent, pour le processus d'ajustement en cours.
Ce jeu d'interactions entre des échecs sortants (E↑) et entrants (E↓) met en évidence le rôle de la coopération implicite entre serveurs dans la convergence du processus d'ajustement. Au lieu de propager des échecs, il peut y avoir neutralisation de ces derniers.

Nous introduisons la notion de latence pour des périodes d'observation successives.

$Lat(PO_k)_{Si}$

Définition: Nous appellons latence, pour la période d'observation PO_k d'un serveur S_i donné, que nous nommons $Lat(PO_k)_{Si}$, la durée qui existe entre l'arrivée d'un échec entrant (E↓) sur $S_i(PO_k)$ et le départ d'un échec sortant (E↑) de S_i, qui est lié à cette arrivée.

La latence est une mesure temporelle (en unité de temps logique - top_i) traduisant une relation de cause à effet entre deux évènements liés à la notion d'échec sortant (E↑) et entrant (E↓) sur les serveurs.
Nous détaillerons plus loin comment cette notion dépend de la probabilité avec laquelle un serveur peut générer des échecs.

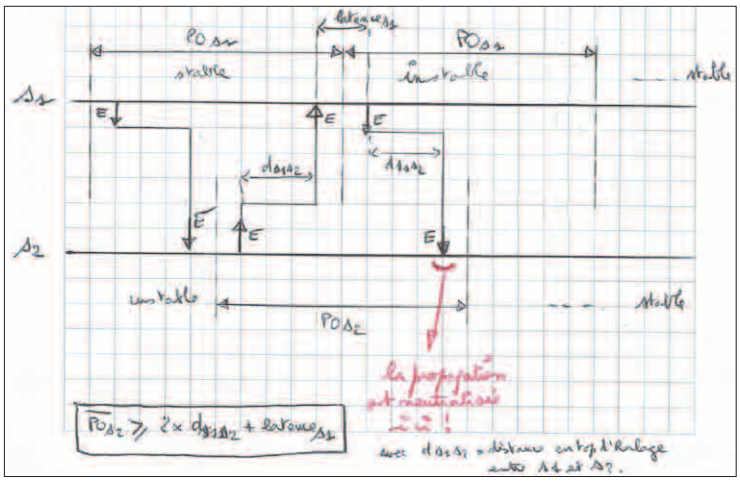


Fig-IV-B4-16

La figure *Fig-IV-B4-16* illustre cette notion de latence et montre son rôle dans la neutralisation des propagations d'échecs.
On voit deux serveurs S_1 et S_2 impliqués dans une interaction implicite, via des requêtes résiduelles dues à des échecs. Chaque serveur étant autonome et ayant sa propre période d'observation, même si ces périodes peuvent avoir une longueur identique 'L' (L_1=L_2=L), elles ne sont, a priori, pas synchronisées.
C'est précisément ce que l'on observe sur la figure *Fig-IV-B4-16*. Le serveur S_1 est stable pendant PO_{1_k}, dans la mesure où l'échec sortant (E↑) se trouve compensé par l'échec entrant (E↓) issu de l'effet provoqué sur S_2 par l'échec sortant (E↑) de S_1. La neutralisation opère car la latence de S_2 (temps de réaction sur une perturbation externe) fait que l'échec propagé arrive avant la fin de PO_{1_k}. En revanche S_2 se trouve alors dans un état instable (échec entrant (E↓) et échec sortant (E↑) à cheval sur PO_{2_j} et PO_{2_j+1}).

Si S_1 émet, à nouveau vers S_2, un échec sortant (E↑) en début de PO_{1_k+1}, il se met dans un état instable. Comme le montre la figure *Fig-IV-B4-16* l'échec entrant (E↓) correspondant sur S_2 arrive avant la fin de PO_{2_j+1}.
Il est neutralisé par S_2, qui n'émettra pas d'échecs sortants (E↑) en cascade. La propagation entre les deux serveurs est ainsi neutralisée et ces derniers deviennent stables.

Cet exemple nous montre l'importance des distances spatio-temporelles, mesurées en nombre de top_i (horloge logique du moteur d'*oRis*), dans la convergence du processus de stabilisation du système.

La neutralisation opère si la longueur de PO_{2_j+1} (L_2) est supérieure ou égale à deux fois la distance (en top_i) entre S_1 et S_2 (d_{S1S2}) plus la latence (toujours en top_i) de S_2 ($Lat()_{S2}$).

Cette condition est nécessaire mais pas suffisante. Il faut, en outre, que cet intervalle temporel soit encadré par une période d'observation.

Il en est de même, de façon croisée, pour S_2 vis-à-vis de S_1 :

$L_1 \geqslant 2xd_{S1S2}+Lat_{S2}$, *(30)*

$L_2 \geqslant 2xd_{S1S2}+Lat_{S1}$. *(31)*

Définition : On appelle distance au prochain échec sortant (E↑) d'un serveur S_i impliqué dans une interaction avec un serveur S_j, que l'on nommera dPE_{i_j}, deux fois la distance (d_{SiSj}) entre S_i et le serveur S_j, impliqué dans ces échecs, plus la latence ($Lat()_{sj}$) de S_j.

dPE_{i_j}

Ces formules *(30)* et *(31)* sont un exemple de relations qui existent entre les positions des serveurs et les longueurs des périodes d'observation.
Notre exemple montre l'interaction entre deux serveurs; dans les simulations un plus grand nombre de serveurs peuvent interagir et provoquer des propagations d'échecs croisées. Pour modéliser correctement ce phénomène, de neutralisation par coopération implicite incluant la notion de latence, il faudrait travailler sur les probabilités qu'un évènement arrive dans un environnement à très forte interaction.
Cette première approche a le mérite de montrer le rôle du positionnement des serveurs entre eux vis-à-vis de leur propre période d'observation respective, dans le processus de recherche d'un état stable global du système.
Cela nous indique également que le déplacement des serveurs leur permet de jouer directement sur le processus. En effet ils auront soit l'attitude de modifier la longueur L_k de leur période d'observation PO_k, soit de se rapprocher ou de s'éloigner de tel ou tel serveur, ou encore de jouer sur les deux à la fois.

Nous avons ici un exemple manifeste de l'importance de la mobilité dans les processus cognitifs avancés (convergence d'un système dynamique vers un état de stabilité global). Nous allons approfondir ces notions dans la partie concernant la synchronisation des serveurs. Un champ d'étude très important s'ouvre alors, et pour lequel des outils de simulation tel qu'*oRis*, que nous verrons dans la partie IV-C, apportent une aide précieuse pour sa compréhension.

Probabilités de générer des échecs sortants (E↑) sur top_t

Nous allons approfondir la gestion des échecs par l'étude de la synchronisation implicite entre les serveurs. Le schéma général, présenté sur la figure *Fig-IV-B4-17*, montre 'p' clients C_i avec trois serveurs S_1, S_2 et S_3. L'ajout de serveurs ne change rien dans le principe.

Une forme d'asynchronisme contextuel complique le processus de recherche de stabilité du système.

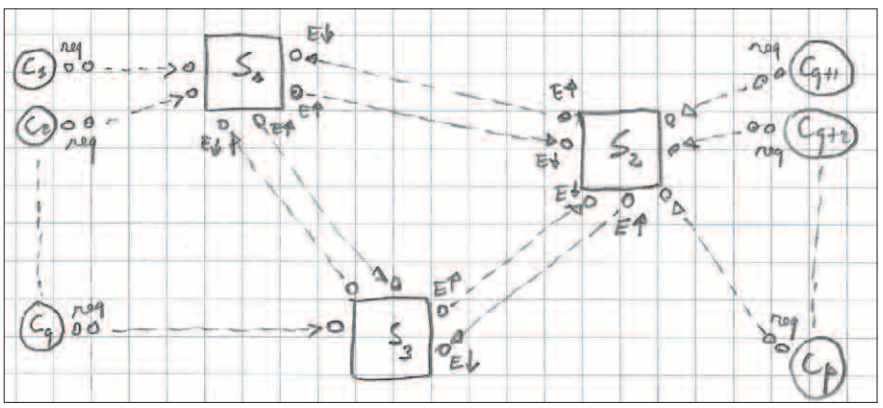


Fig-IV-B4-17

Les top_i logiques de la simulation rendent bien compte de ce phénomène. Ceci est démultiplié par l'usage d'agents requête qui sont mobiles.
Nous verrons dans la partie IV-D (*oRis* et nos simulations), comment cette horloge logique, centralisée au niveau du processus hébergeur, nous permettra, paradoxalement, de donner aux agents des comportements autonomes et désynchronisés.
En réalité ces techniques, portées par ce que l'on appelle les systèmes multi-agents, ne sont pas complètement nouvelles. On retrouve des principes analogues dans les ordonnanceurs des systèmes d'exploitation dont la principale mission est de faire avancer tous les processus d'une machine dans une forme de parallèlisme anthropomorphique, ou ressenti.

Le paradoxe vient du fait que la synchronisation de tous les agents sur une horloge logique partagée, et donc centralisée par construction du similateur, permet à chacun de « vivre sa vie » indépendemment des autres.

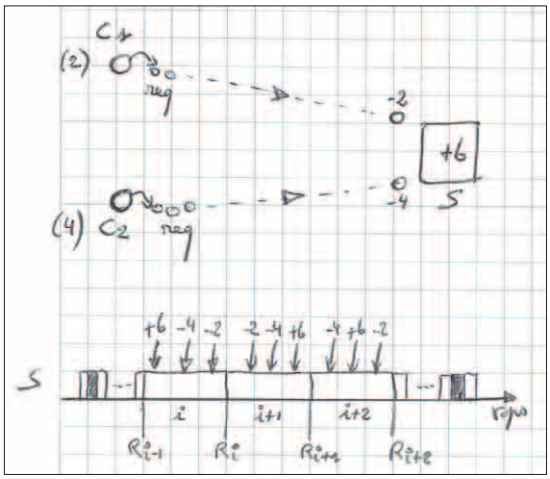


Fig-IV-B4-18

Sans préjuger des instants dans lesquels les requêtes ont été émises par les clients C_1 et C_2 sur la figure *Fig_IV-B4-18*, regardons ce qui se passe lorsqu'elles arrivent à proximité du serveur visé 'S'.
La requête, qui vient de C_1, est porteuse d'une demande de valeur 2, alors que celle qui arrive de C_2 a une valeur de quatre unités de ressources. Le schéma, de la figure *Fig_IV-B4-18*, montre l'évolution de la réserve 'R' du serveur à chaque top_i d'une période d'observation PO_k donnée.

En fait, dans ce cas, trois évènements concurrents et indépendants se produisent à chaque top_i : la production p_i du serveur augmente la valeur de sa réserve R_{i-1}, alors que le traitement des deux requêtes (respectivement -2 et -4) diminue cette réserve. Ces évènements, bien qu'arrivant tous les trois à chaque top_i, peuvent être pris en compte de différentes façons suivant l'ordre qui sera choisi par le serveur (ou plus exactement l'ordonnanceur interne au simulateur) parmi toutes les possibilités ((n+1)! pour 'n' clients et un serveur).

Toujours sur la figure *Fig-IB-B4-18*, on voit que pendant le top_i il n'y a pas d'échec sortant (E↑) émis, et ce quelque soit l'état de la réserve; alors que pour les top_{i+1} et top_{i+2} il peut y en avoir en fonction de l'état de la réserve, car la production arrive après les demandes. Dans ce cas, nous aurions la somme des demandes qui dépasserait la réserve au début des top_{i+1} et top_{i+2}, provoquant l'émission d'échecs.

Certains outils permettent de lever ce problème en stockant tous les évènements à venir, pour un top_i donné, et en les traitant de façon atomique.
C'est le cas de plate-formes synchrones, comme *MOOREA*, qui garantissent l'ordre d'exécution d'évènements dans un top d'horloge logique, respectant l'ordre d'arrivée des dits évènements dans le top.

Encore faut-il, dans ce cas, ordonner les évènements avant de les traiter pour minimiser les cas d'échec.

C'est ce qui arrive dans *UNIX* avec l'algorithme de tri des blocs physiques, avant lecture sur un disque externe ; dans ce cas la notion d'échec correspond aux allers-retours inutiles du bras de lecture.

On illustre l'apparition d'échecs sur la figure *Fig-IV-B4-19* en précisant la valeur de la réserve en début de chaque top_i. Ici elle vaudra 5.

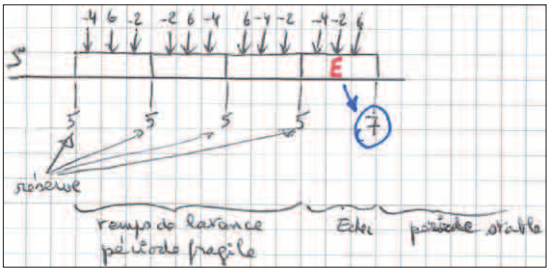


Fig-IV-B4-19

La latence correspond à une période pendant laquelle la probabilité de générer un échec sortant (E↑), consécutivement à l'arrivée d'un échec entrant (E↓), est faible. C'est uniquement lors du quatrième top_i que l'ordre du traitement des évènements conduit à l'échec sortant (E(-2)↑).

En effet (-4) et (-2) correspondent à une demande de six unités de ressources, alors que la réserve est de cinq. La deuxième demande (-2) ne sera pas traitée par ce serveur et sa réserve en fin de top_i sera de sept au lieu de cinq dans la période stable.

Soit PE_{i_k} la probabilité de générer un échec sortant (E↑) pour un top_i donné, d'une période d'observation PO_k. Cette probabilité PE_{i_k} dépend des demandes à traiter sur top_i, mais aussi de la réserve res_i en début de top_i.

PE_{i_k}

Nous avons le cas (a) où cette réserve res_i est supérieure ou égale aux demandes à venir dans top_i, et le cas (b) où elle ne l'est pas :

- *(a)* : ici la probabilité PE_{i_k} de générer un échec est nulle quelle que soit la production p_i, alors que dans le second cas (b) cela va dépendre de l'ordre d'exécution des évènements (et/ou de leur arrivée) et de p_i. Sur la figure *Fig-IV-B4-20* nous voyons que quel que soit l'ordre d'arrivée des évènements aucun échec ne sera généré (PE_{i_k}=0) ;
- *(b)* : en effet ici il y a un risque de générer des échecs puisque la somme des demandes est supérieure à la réserve en début de top_i.

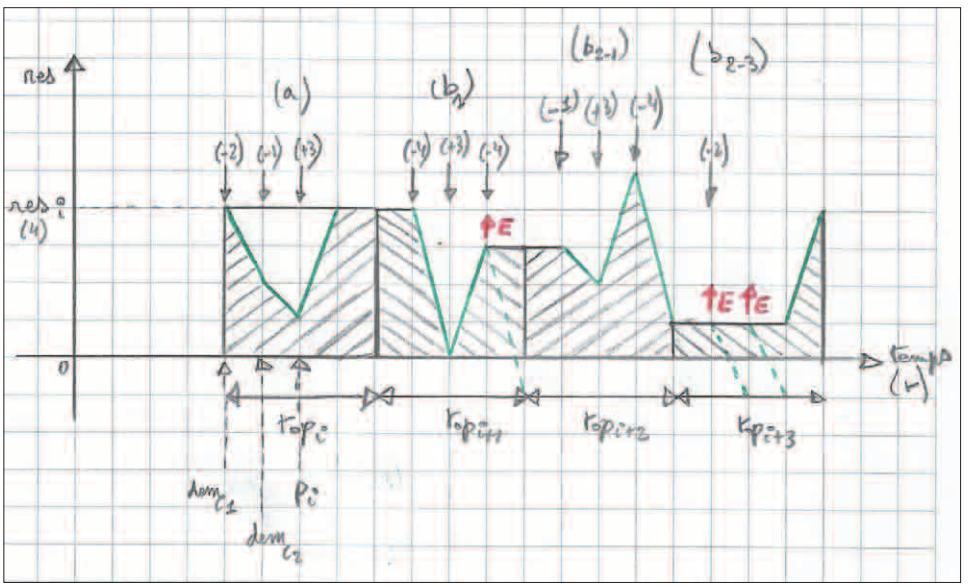


Fig-IV-B4-20

Plusieurs cas sont à prendre en compte :

- *(b_1)* : si la somme des demandes est supérieure au potentiel (réserve+production), alors la probabilité PE_{i_k} de générer un échec vaut 1. C'est le cas de top_{i+1} sur la figure *Fig-IV-B4-20*, où les deux demandes ((-4) et (-4)) dépassent la somme de la réserve (+4) plus la production (+3) ;
- *(b_2)* : si la somme des demandes est inférieure ou égale au potentiel (réserve+production), alors la probabilité PE_{i_k} de générer un échec dépend de l'ordre des évènements. C'est le cas de top_{i+2} et top_{i+3} sur la figure *Fig-IV-B4-20*.

Nous pouvons encore distinguer plusieurs sous-cas :

- *cas (b_{2_1})* : ici la réserve est strictement inférieure à la somme des demandes. Il faut trouver (numérateur de PE_{i_k}) le nombre de possibilités d'avoir toutes les demandes avant de produire localement de nouvelles ressources. Le numérateur vaut (n!).
  Nous obtenons donc une probabilité PE_{i_k} :
  $PE_{i\_k}=n!/(n+1)!=1/(n+1)$, avec $\lim_{n\to\infty} PE_{i\_k}=0$. *(32)*
  Sur l'exemple de la figure *Fig-IV-B4-20* cela donne $PE_{i\_k}=⅓$;
- *cas (b_{2_2})* : ici la réserve est strictement inférieure à exactement l'une des demandes. Il faut trouver (numérateur de PE_{i_k}) le nombre de possibilités de n'avoir que cette demande en premier évènement.
  Le numérateur *num* s'exprime de la façon suivante :
  $num=1/2[(n+1)!-2n!]+n!=1/2(n+1)!$
  Nous obtenons donc une probabilité PE_{i_k} :
  $PE_{i\_k}=[1/2(n+1)!]/(n+1)!=½$. *(33)*
- *cas (b_{2_3})* : enfin ici la réserve est strictement inférieure à toutes les demandes prises séparément. Il faut trouver le nombre de possibilités (numérateur de PE_{i_k}) de ne pas avoir la production en premier évènement.
  Le numérateur *num* s'exprime de la façon suivante :
  $num=(n+1)!-n!=n(n!)$.
  Nous obtenons donc une probabilité PE_{i_k} :
  $PE_{i\_k}=[n(n!)]/(n+1)!=n/(n+1)$, *(34)*
  avec $\lim_{n\to\infty} PE_{i\_k}=1$.
  Sur l'exemple de la figure *Fig-IV-B4-20* cela donne $PE_{i\_k}=⅔$.

Pour ne pas alourdir les formules, nous n'avons pas pris les cas entre *(b_{2_2})* et *(b_{2_3})* qui consistent à avoir une réserve qui est strictement inférieure à plus d'une demande.
Le cas extrême (la réserve est inférieure à toutes les demandes) revient à *(b_{2_3})*.

Nous pouvons commenter ces résultats sur l'exemple de la figure *Fig-IV-B4-20*. Le top_{i+2} appartient au cas *(b_{2_1})*. Bien que $PE_{i+2\_k}=⅓$, la situation choisie ne génère pas d'échec sortant (E↑). En effet la production p_{i+2} arrive au milieu des demandes.

Le top_{i+3} appartient lui au cas *(b_{2_3})*. Dans la mesure où la réserve res_{i+3} est inférieure à toutes les demandes prises séparément et que ces dernières arrivent avant la production p_{i+3}, nous avons autant d'échecs sortants (E↑) potentiels que de demandes.

Nous voyons sur cet exemple que la probabilité de provoquer au moins un échec sortant (E↑) est de ⅔, mais que le nombre d'échecs engendrés peut aller d'un à 'n' suivant l'ordre des évènements.
Dans le cas (b_{2_3}), nous pouvons exprimer les probabilités d'engendrer 'p' échecs, que nous noterons $PE_{i\_k}(p)$, pour 'p' compris entre 0 et 'n'. La valeur 0 correspond à une production en début de top_i, donc sans échec sortant (E↑) généré.

$$PE_{i\_k}(p)=n!/[(n+1)!]=1/(n+1),\ \forall p\in[0,n]. \qquad (35)$$

En effet nous avons :

$$PE_{i\_k}(0)+PE_{i\_k}(1)+\ldots+PE_{i\_k}(n)=\sum_{i=0}^{n}[1/(n+1)]=[(n+1)/(n+1)]=1.$$

Cela signifie que nous avons la même probabilité de générer un, deux ou 'n' échecs. On voit bien que plus 'n' est grand et plus la probabilité de ne pas obtenir d'échec est faible (1/(n+1)), alors que celle d'en obtenir au moins un vaut (1-(1/(n+1))) et tend vers 1 :

$$\lim_{n\to\infty}[1-(1/(n+1))]=1.$$

Nous verrons dans les simulations que plusieurs options sont possibles pour gérer les échecs sortants (E↑). Lorsqu'un échec sortant (E↑) se produit, cela signifie qu'un agent requête a échoué auprès d'un serveur indisponible par manque de ressources. Dans ce cas la requête doit se mettre en recherche d'un autre serveur. Elle devient une requête résiduelle (cf. la figure *Fig-IV-B4-21*).
On peut facilement imaginer qu'une telle requête n'arrive pas à trouver un serveur disponible ou reviennent vers un serveur déjà visité pour retenter sa chance.

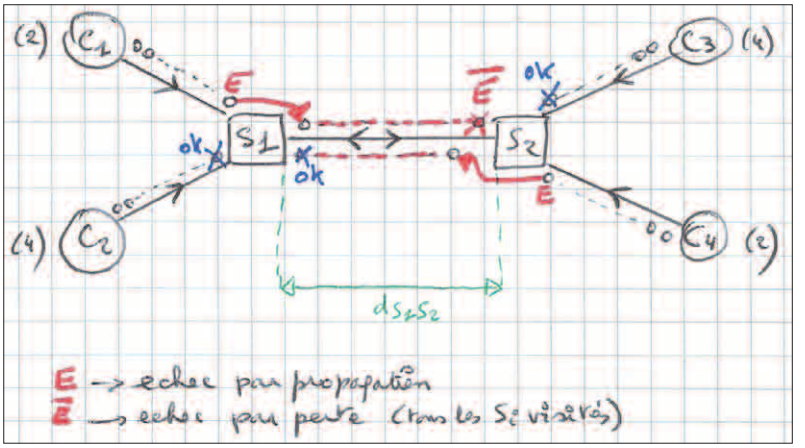


Fig-IV-B4-21

Plusieurs stratégies sont alors possibles.
Les agents requête peuvent mémoriser les serveurs visités et s'interdire, ou non, de revenir sur un agent déjà visité. Ils peuvent également se donner une durée de vie au-delà de laquelle ils disparaissent pour ne pas chercher indéfiniment un serveur. On parle, dans ce cas, d'échec par perte (cf. la figure *Fig-IV-B4-21*).
La demande correspondante ne sera jamais satisfaite. Une mesure d'efficacité des heuristiques consiste à minimiser ces pertes.

Sans anticiper sur la suite des travaux, nous verrons que ces agents requêtes vont jouer un rôle décisif dans les heuristiques, au-delà de satisfaire les besoins des clients en recherchant les serveurs disponibles.
En effet, et à l'image des phéromones chez les fourmis, leur exploration systématique du système permet de mettre à jour, en temps réel, les connaissances des différents acteurs du dispositif sur l'état du système global.

L'introduction du modèle d'interaction, appelé la *dérive des connaissances* (proposé dans ma thèse), s'appuiera sur ce travail exploratoire. Il permettra, entre autres, aux serveurs de créer des liens relationnels entre eux, facilitant les résolutions coopératives.

Probabilités de générer spontanément des échecs sortants (E↑) sur PO_k

Prenons le cas d'une période d'observation PO_k pendant laquelle le serveur S_i, correspondant, ne reçoit pas les demandes de façon homogène sur l'ensemble des L_k unités temporelles (top_i).

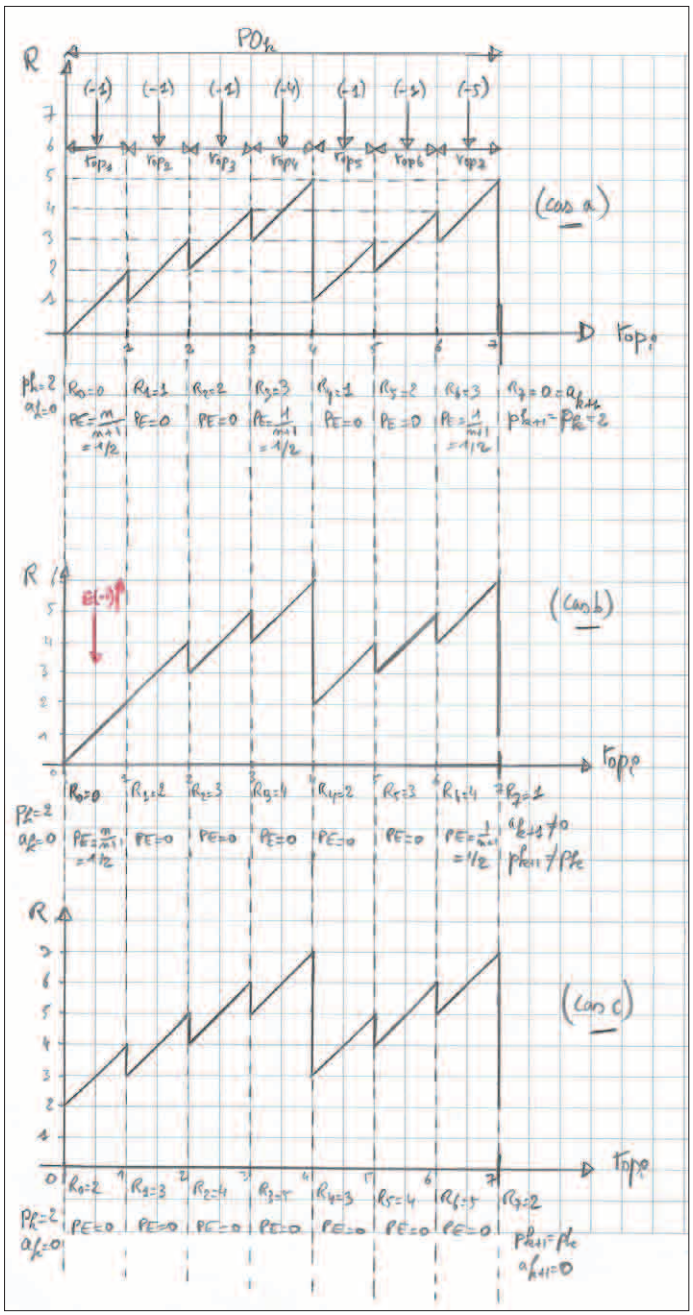


Fig-IV-B4-22

Soit l'exemple de la figure *Fig-IV-B4-22* (cas 'a', 'b' et 'c') où PO_k est composé de sept top_i (L_k=7 et i∈[1,7]). Sur chacun d'eux arrivent, en provenance des clients, les demandes (une par top_i) externes suivantes:

$dem(top_1)=dem(top_2)=dem(top_3)=$
$dem(top_5)=dem(top_6)=-1$,
$dem(top_4)=-4$ et
$dem(top_7)=-5$.

Détaillons ces cas, à partir de la figure *Fig-IV-B4-22*, où la probabilité PE_{i_k} de générer un échec sortant (E↑) pendant le top_i de PO_kest notée PE:

- cas 'a': ces probabilités montrent que seuls top_1, top_4 et top_7 sont à risque. Sur notre exemple elles valent ½ ($PE_{1\_k}=PE_{4\_k}=PE_{7\_k}=½$), puisque le nombre de demandes dans chaque top_i est de 1 (n=1). Pour les autres top_i nous avons $PE_{i\_k}=0$.

On voit que le risque de générer des échecs dépend de la répartition des demandes sur les différents top_i de PO_k. Nous étudierons, juste après, le cas d'une répartition homogène des demandes sur PO_k.

Au départ de PO_k (top_1) la réserve et l'ajustement sont nuls ($R_0=a_k=0$) et la production vaut 2 ($p_k=2$);

- cas 'b': les conditions sont identiques à celles du cas précédent, mais il y a un échec sortant (E(-1)↑) avec $PE_{1\_k}=½$, qui est généré pendant la première unité temporelle (top_1). Cela décale la courbe de la réserve 'R' d'une unité. La conséquence de cela se voit sur les probabilités PE_{i_k} pour les top_i suivants. En effet la probabilité à top_4 passe de ½ à 0, rendant improbable la génération des échecs sortants (E↑), alors que celle à top_7 reste égale à ½. Puisque R_7=1 en fin de

PO_k, l'ajustement fonctionne pour PO_{k+1} et nous avons un état instable avec $a_{k+1} \neq a_k$ et $p_{k+1} \neq p_k$. La génération d'échecs sortants (E↑) en début de PO_k diminue les chances d'en avoir en fin de PO_k, raccourcissant d'autant la latence ($Lat(PO_k)$) de PO_k. Ceci est au prix de rendre instable la période suivante;

- cas 'c' : ici nous partons avec une réserve non nulle, qui est égale à la production ($R_0=p_k$). Plus aucun top_i de PO_k ne peut générer des échecs sortants (E↑), puisque $PE_{i\_k}=0$.

Regardons sur la figure *Fig-IV-B4-23* le cas d'une période d'observation PO_k pendant laquelle le serveur S_i correspondant reçoit des demandes de façon homogène sur l'ensemble des L_k unités temporelles (top_i) :

- cas 'a' : cette fois nous avons une demande de deux unités de ressource (-2) pour chaque top_i, qui est alignée sur la production ($p_k=2$). La probabilité PE_{i_k} pour chaque top_i vaut ½. Il y a donc une chance sur deux de générer un échec sortant (E↑) par top_i;
- cas 'b' : nous voyons, au top_2, cas 'b', sur la figure *Fig-IV-B4-23*, que la génération d'un échec sortant (E↑) décale la courbe 'R' de deux unités pour les top_i suivants. Là encore plus l'échec sortant (E↑) arrive vite dans PO_k et plus la latence ($Lat(PO_k)$) sera courte. Ce qui est intéressant pour accélérer la convergence du processus de stabilisation;
- cas 'c' : il confirme l'utilité d'avoir une réserve R_i ègale à la production p_k, puisque cela bloque la génération d'échecs sortants (E↑).

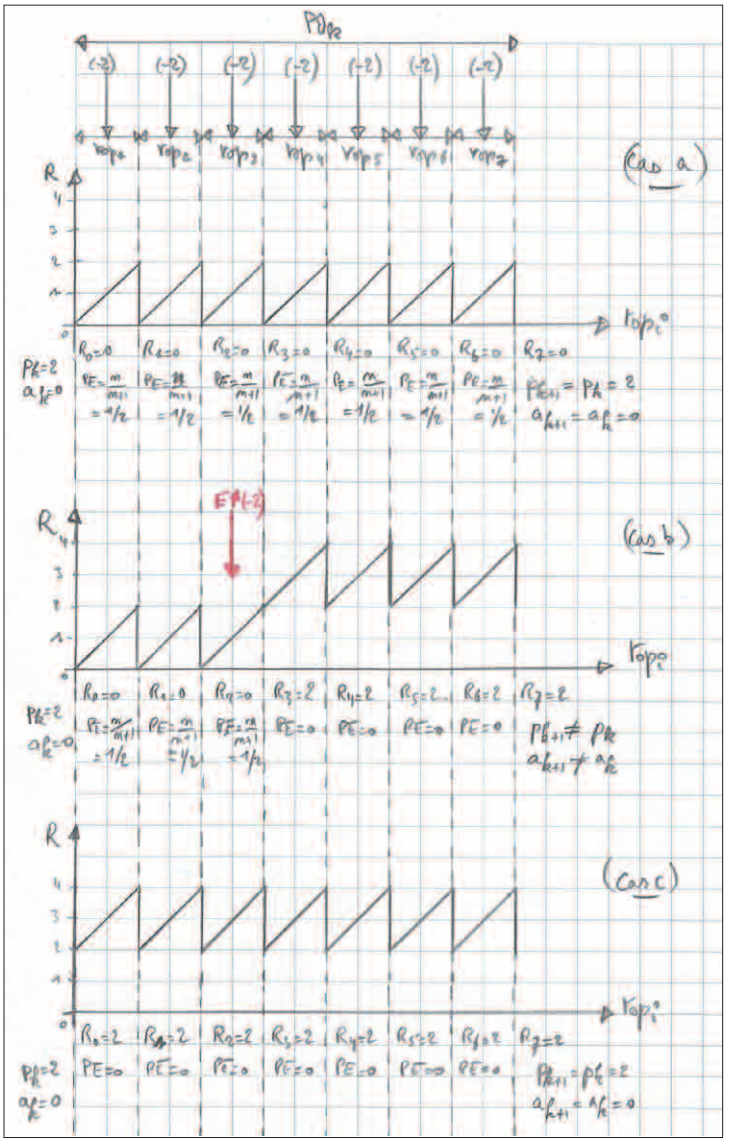


Fig-IV-B4-23

Dans ces deux derniers exemples nous avons considéré une demande (n=1) par unité temporelle logique (top_i) et la possibilité d'avoir des échecs sortants (E↑) suivant la probabilité PE_{i_k}.
Soit cela représente une demande moyenne, elle-même constituée de plusieurs demandes, on aura alors $PE_{i\_k}=1/(n+1)$, soit on peut effectivement considérer 'n' demandes, dans ce cas on aura $PE_{i\_k}=n/(n+1)$.

Dans le premier cas, lorsque 'n' est grand, PE_{i_k} tend vers 0, alors que dans le second cas, plus 'n' est grand et plus PE_{i_k} tend vers 1.

Calcul de la latence de PO_k sur le serveur S_i

Nous allons regarder comment calculer la latence pour la période d'observation PO_k sur un serveur S_i, afin de mieux comprendre les règles comportementales qui pourraient optimiser le processus d'ajustement.

Rappel : La latence correspond à une durée, exprimée en unité temporelle logique (top_i), qui sépare l'arrivée d'un échec entrant (E↓) du départ d'un échec sortant (E↑), consécutif à cette arrivée. Cela mesure le degré de réactivité d'un serveur à un évènement imprévu (hors ajustement).

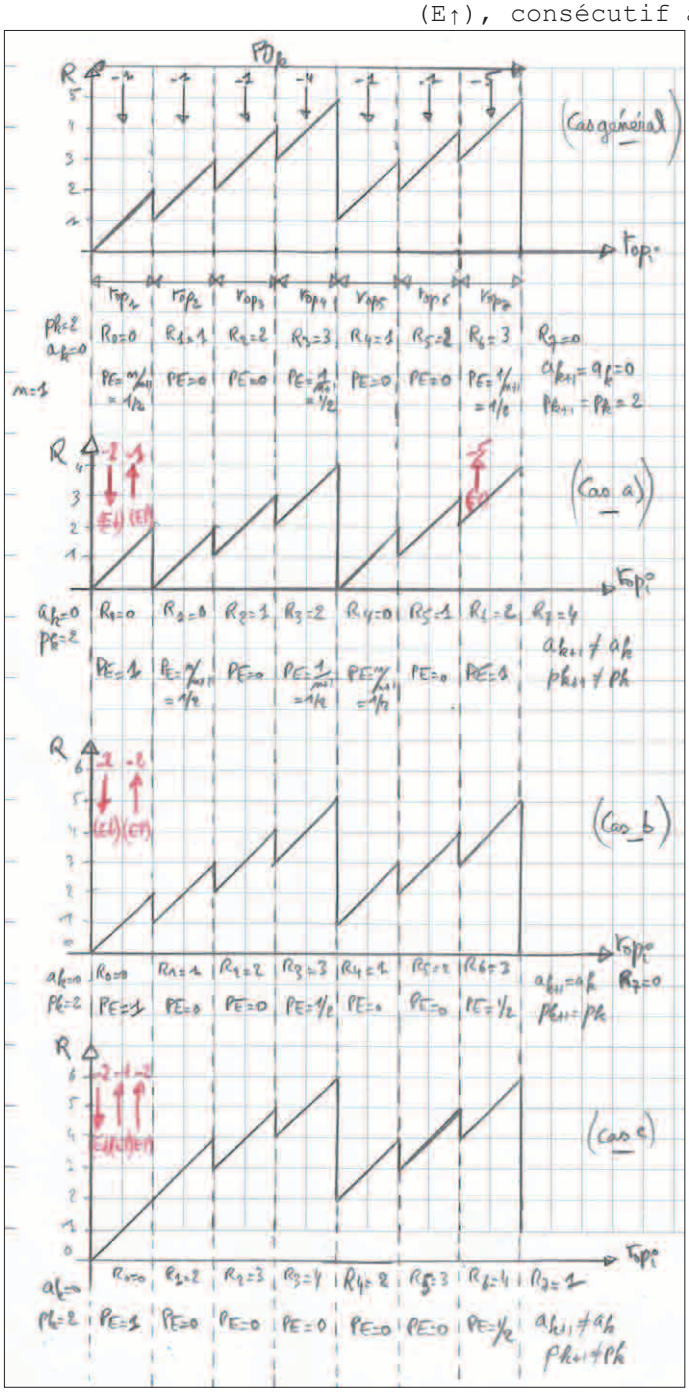


Fig-IV-B4-24

Nous allons voir qu'il est intéressant de définir deux latences différentes et complémentaires (Lat_m et Lat_M). Avant cela prenons un exemple à partir duquel nous allons construire les formules générales.

Nous avons, sur la figure *Fig-IV-B4-24*, un serveur S_i et sa période d'observation PO_k de longueur L_k=7.

Il arrive une seule demande (n=1) par top_i dont les valeurs valent respectivement -1 pour les top_1, top_2, top_3, top_5, top_6, -4 pour le top_4 et et -5 pour le top_7.

Le cas général, représenté sur la figure *Fig-IV-B4-24*, reprend les probabilités de générer spontanément un échec sortant (E↑), comme nous l'avons vu précédemment. Dans ce cas général, lorsqu'un échec entrant (E(-2)↓) arrive sur top_1, alors que la réserve est nulle (R_0=0), l'ajustement aussi (a_k=0) et la production constante (p_k=2), cela conduit à trois évènements (n=3) qui sont la demande classique (-1), l'échec entrant (E(-2)↓) et la production (+2).

Il existe n! possibilités, ici 3!=6, pour ordonner ces évènements.

Ce qui est remarquable dans notre exemple (demande classique (-1), échec entrant (E(-2)↓) et production (+2)), c'est que quelque soit l'ordre de ces trois évènements, il y aura génération d'au moins un échec sortant (E↑).

En effet $R_0+p_k<|\alpha+dem_k|$ avec R_0=0, p_k=2, α=-2 et dem_k=-1. Dans cette configuration on aura PE_{1_k}=1 et donc $Lat(PO_k)_{Si}$=1 avec une probabilité qui vaut 1.

Mais si l'on regarde de plus près les différentes combinaisons d'évènements, nous avons :

c_1={-2,-1,2} ⇒(E(-2)↑) et (E(-1)↑),
c_2={-2,2,-1} ⇒(E(-2)↑),
c_3={-1,-2,2} ⇒(E(-1)↑) et (E(-2)↑),
c_4={-1,2,-2} ⇒(E(-1)↑),
c_5={2,-1,-2} ⇒(E(-2)↑),
c_6={2,-2,-1} ⇒(E(-1)↑).

Suivant ces combinaisons (c_i), on observe la génération d'au moins un échec sortant (E↑) avec :
- possiblement davantage, c'est le cas pour c_1 et c_3 ;
- et des valeurs associées aux échecs, différentes (-2 pour c_2 et c_5, et -1 pour c_4 et c_6).

Cela donne, sur la figure *Fig-IV-B4-24*, les trois cas *(a)*, *(b)* et *(c)*, correspondant respectivement aux combinaisons c4/c6, c2/c5 et c1/c3 avec la probabilité PE=⅓ pour chacun de ces trois cas :

PE(E(-1)↑)=PE(E(-2)↑)=PE((E(-1)↑) & (E(-2)↑))=⅓ et
PE(E(-1)↑)+PE(E(-2)↑)+PE((E(-1)↑) & (E(-2)↑))=1.

Dans le cas de la définition de la latence, telle qu'elle a été introduite en amont, ces trois cas ne changent rien, puisque de façon sûre au moins un échec sortant (E↑) sera généré à l'issue de top_1.
On pourra dire que $Lat(PO_k)_{Si}$=1 sera réalisé avec une probabilité de 1.

Cela signifie que la distance au prochain échec sortant (E↑) consécutivement à un échec entrant (E↓) sera d'une unité (top_i logique), traduisant ici une très grande réactivité.

Regardons sur ces trois cas, les probabilités de générer un ou plusieurs échecs sortants (E↑) consécutivement à cet évènement :
- *cas 'a'* : l'échec sortant (E↑) de top_1 vaut -1. Cela donne PE=½ pour top_2, top_4 et top_5, et surtout PE_{7_k}=1 pour top_7, c.-à-d. qu'un deuxième échec sortant (E(-5)↑) est généré de façon sûre. Sa valeur étant importante (-5), le reste R_7 va être important (4), rendant S_i instable (a_{k+1}≠0 et p_{k+1}≠p_k) ;
- *cas 'b'* : ici l'échec sortant (E↑) de top_1 vaut -2, ce qui compense et annule l'effet de l'échec entrant (E↓) sur top_1. Cela entraîne un retour à la situation de départ («cas général» sur la figure *Fig-IV-B4-24*) ;
- *cas 'c'* : ce dernier cas conduit à la génération de deux échecs sortants (E↑) sur top_1 de valeurs respectivement -1 et -2, renforçant de façon significative la réserve (+1 à chaque top_i). La conséquence est une diminution du nombre des top_i dont PE_{i_k}≠0, avec S_i instable (a_{k+1}≠0 et p_{k+1}≠p_k) pour la période suivante.

Définition : Nous appellons $Prob(Lat(PO_k)_{Si}=p)$ la probabilité avec laquelle $Lat(PO_k)_{Si}=p$. $Prob(Lat(PO_k)_{Si}=p))$

Dans l'exemple de la figure *Fig-IV-B4-24*, pour les cas *(a)*, *(b)* et *(c)* nous avons les probabilités suivantes :

$Prob(Lat(PO_k)_{Si}=1)=1$,
$Prob(Lat(PO_k)_{Si}=p)=0$ pour p∈[2,7].

Il nous faut affiner la notion de latence telle qu'elle a été définie précédemment, afin de prendre en compte les phénomènes observés sur notre exemple. En effet nous allons considérer la latence minimale (Lat_m) et la latence maximale (Lat_M).

$Lat_m(PO_k)_{Si}$ *Définition*: La latence minimale, notée $Lat_m(PO_k)_{Si}$, correspond à ce que nous appelions jusque-là la latence. Il s'agit donc, pour la période d'observation PO_k d'un serveur S_i donné, de la durée qui existe entre l'arrivée d'un échec entrant (E↓) sur $S_i(PO_k)$ et le départ du premier échec sortant (E↑) de S_i, qui est lié à cette arrivée.

$Lat_M(PO_k)_{Si}$ *Définition*: La latence maximale, notée $Lat_M(PO_k)_{Si}$, correspond, pour la période d'observation PO_k d'un serveur S_i donné, à la durée qui existe entre l'arrivée d'un échec entrant (E↓) sur $S_i(PO_k)$ et le départ du dernier échec sortant (E↑) de $S_i(PO_k)$, qui est lié à cette arrivée.

Ces deux notions permettent de définir un intervalle de réactivité d'un serveur à un échec entrant (E↓).

LATENCE MINIMALE $Lat_m(PO_k)_{Si}$

Nous allons commencer par proposer les formules générales du calcul des probabilités pour la latence minimale ($Prob(Lat_m(PO_k)_{Si}=p)$) avec $p\in[1,L_k]$.

Prenons le cas général de la figure *Fig-IV-B4-24*; nous avons les trois configurations suivantes:

- cas 'a':
  $PE_{1\_k}=1$, donc $Prob(Lat_m(PO_k)_{Si}=1)=PE_{1\_k}=1$;
  dans ce cas un déséquilibre s'installe puisque l'échec sortant (E↑) n'est pas équivalent en valeur à l'échec entrant (E↓);
- cas 'b':
  $PE_{1\_k}=1$, donc $Prob(Lat_m(PO_k)_{Si}=1)=PE_{1\_k}=1$;
  dans ce cas aucun déséquilibre s'installe puisque l'échec sortant (E↑) est équivalent en valeur à l'échec entrant (E↓);
- cas 'c':
  $PE_{1\_k}=1$, donc $Prob(Lat_m(PO_k)_{Si}=1)=PE_{1\_k}=1$;
  même cas qu'en 'a', un déséquilibre s'installe puisque les échecs sortants (E↑) ne sont pas équivalents en valeur à l'échec entrant (E↓).

Pour trouver les valeurs suivantes nous devront appliquer la règle suivantes: la probabilité P() que 2 évènements 'E' et 'F', indépendants, se réalisent, est le produit des probabilités de chacun de ces évènements:

$$P(E \& F)=P(E)xP(F). \qquad (36)$$

Donc pour calculer $Prob(Lat_m(PO_k)_{Si}=2)$, l'évènement 'E' correspond à $\neg Prob(Lat_m(PO_k)_{Si}=1)=1-Prob(Lat_m(PO_k)_{Si}=1)$ et l'évènement 'F' correspond à PE_{2_k}.

Nous obtenons:

$$Prob(Lat_m(PO_k)_{Si}=2)=[1-Prob(Lat_m(PO_k)_{Si}=1)]xPE_{2\_k}.$$

Sur notre exemple cela donne :

$Prob(Lat_m(PO_k)_{Si}=2)=[1-1]x0=0$.

En effet la probabilité de générer un échec sortant (E↑) sur top_2 étant nulle, il n'y a aucune chance pour que la latence minimale de S_i sur PO_k soit égale à 2.

Au-delà de cela, la probabilité d'avoir $Lat_m(PO_k)_{Si}=1$ étant toujours vrai, la négation sera toujours nulle, donc :

$\forall j \in [2,L_k]$, $Prob(Lat_m(PO_k)_{Si}=j)=0$.

Pour généraliser cette formule il faut considérer le rang 'p' qui correspond au top_p de PO_k, avec $p \in [1,L_k]$, à partir duquel un échec sortant (E↑) est généré dans PO_k consécutivement à l'arrivée d'un échec entrant (E↓) sur la même période.

Si l'échec entrant (E↓) arrive sur top_1 de PO_k, alors nous obtenons la formule *(37)* suivante $\forall p>1$:

$$Prob(Lat_m(PO_k)_{S_i}=p)=\left[\prod_{j=1}^{p-1}(1-Prob(Lat_m(PO_k)_{S_i}=j))\right]\times PE_{p\_k} \qquad (37)$$

$$\textbf{avec } Prob(Lat_m(PO_k)_{S_i}=1)=PE_{1\_k}$$

Si l'échec entrant (E↓) n'arrive pas sur top_1 (le 1er top de PO_k), mais sur le le avec $l \in [2,p]$, alors nous aurons :

$$Prob(Lat_m(PO_k)_{S_i}=\alpha)=\left[\prod_{j=l}^{p-1}(1-Prob(Lat_m(PO_k)_{S_i}=j))\right]\times PE_{p\_k} \qquad (38)$$

$$\textbf{et } \alpha=p-l+1$$

$$\textbf{avec } Prob(Lat_m(PO_k)_{S_i}=l)=PE_{l\_k}$$

Dans le cas général de notre exemple, vu sur la figure *Fig-IV-B4-24*, que nous avons reporté sur la figure *Fig-IV-B4-25*, nous obtenons pour la latence minimale Lat_m les résultats suivant les trois cas (d_1), (d_2) et (d_3) :

- cas (d_1) : l'échec entrant (E(-1)↓) arrive sur top_2,
  $Prob(Lat_m(PO_k)_{Si}=1)=½$,
  $Prob(Lat_m(PO_k)_{Si}=2)=(1-(½))x0=0$,
  $Prob(Lat_m(PO_k)_{Si}=3)=(1-(½))x(1-0)x(½)=¼$,
  $Prob(Lat_m(PO_k)_{Si}=4)=(1-(½))x(1-0)x(1-(¼))x(½)=3/16$,
  $Prob(Lat_m(PO_k)_{Si}=5)=(1-(½))x(1-0)x(1-(¼))x(1-(3/16))x0=0$,
  $Prob(Lat_m(PO_k)_{Si}=6)=(1-(½))x(1-0)x(1-(¼))x(1-(3/16))x(1-0)x1=39/128$;

- cas (d_2) : l'échec entrant (E(-1)↓) arrive sur top_3,
  $Prob(Lat_m(PO_k)_{Si}=1)=0$,
  $Prob(Lat_m(PO_k)_{Si}=2)=(1-0)x(½)=½$,
  $Prob(Lat_m(PO_k)_{Si}=3)=(1-0))x(1-(½))x(½)=¼$,
  $Prob(Lat_m(PO_k)_{Si}=4)=(1-0)x(1-(½))x(1-(¼))x0=0$,
  $Prob(Lat_m(PO_k)_{Si}=5)=(1-0)x(1-(½))x(1-(¼))x(1-0)x1=⅜$;

- cas (d_3) : l'échec entrant (E(-1)↓) arrive sur top_5,
  $Prob(Lat_m(PO_k)_{Si}=1)=½$,
  $Prob(Lat_m(PO_k)_{Si}=2)=(1-(½))x0=0$,
  $Prob(Lat_m(PO_k)_{Si}=3)=(1-(½))x(1-0)x1=½$.

On voit sur cet exemple que la probabilité d'avoir une latence minimale d'une certaine valeur, dépend de l'endroit (top_i) où arrive l'échec entrant (E↓).

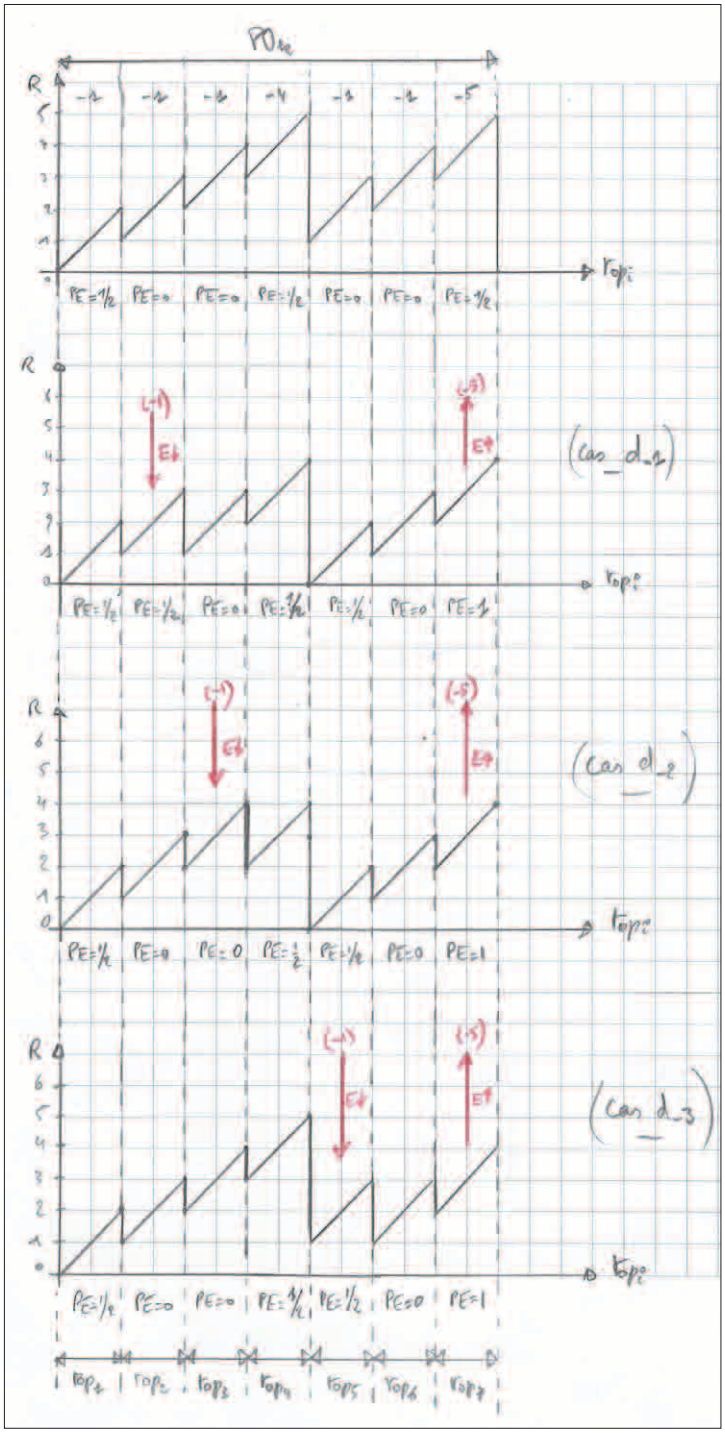


Fig-IV-B4-25

La formule générale *(38)* du calcul de la latence minimale (Lat_m) nous montre que les valeurs possibles pour Lat_m sont directement liées à PE_{i_k}, mais aussi au top_i sur lequel arrive l'échec entrant (E↓).
Si PE_{i_k} est nulle pour top_i, alors Lat_m ne pourra avoir la valeur correspondant à top_i. Sur notre exemple Lat_m=2 ou 3 ou 5 ou 6 n'est pas possible.

L'intérêt des serveurs, pour accélérer la convergence du processus d'ajustement, est de minimiser les latences minimales afin d'augmenter leur réactivité.
Cette réactivité permet d'augmenter les chances de neutralisation des échecs entrants (E↓) et sortants (E↑).

Plus vite un échec sortant (E↑) est produit consécutivement à l'arrivée d'un échec entrant (E↓), plus nous augmentons les chances que cela se passe à l'intérieur de la PO_k courante, avec pour conséquence de ne pas déclencher le processus d'ajustement en cas de stabilité du serveur.

Symétriquement le serveur qui a déclenché une requête résiduelle vers un autre serveur aura une distance au prochain échec (dPE) d'autant plus courte, lui permettant aussi de gérer l'échec entrant (E↓) dans sa période d'observation courante.

On voit que la réactivité diminue les chances de perturbation des processus d'ajustement en cours dans une interaction entre processus.
Cela s'ajoute à l'idée de rapprocher deux serveurs (S_i,S_j) en interaction, dont l'effet est de diminuer leur distante (d_{SiSj}) afin de réduire la distance au prochain échec (dPE).

Nous avons vu que faire tendre la réserve vers la quantité de ressources produite ($R \rightarrow p_k$) minimise les échecs sortants (E↑). C'est particulièrement visible dans le cas de demandes homogènes sur PO_k. En même temps nous venons de voir qu'il est intéressant d'augmenter la probabilité d'avoir une latence minimale la plus courte possible. Or pour cela il faut minimiser la réserve 'R', sinon les échecs sortants (E↑) risquent d'apparaître tardivement, augmentant d'autant la dPE, avec les risques de déstabilisation du processus d'ajustement.
Cette apparente contradiction n'en est pas une, dans la mesure où il s'agit en fait d'adapter son comportement à la situation.

Pour se stabiliser au plus vite dans des périodes sans échecs entrants (E↓), le serveur devra faire tendre sa réserve vers sa production (R→p_k). Si ce n'est pas le cas, il devra privilégier une réserve faible (R→0) afin d'augmenter ses chances d'avoir une latence minimale courte.

Latence maximale $Lat_M(PO_k)_{S_I}$

Comme indiqué précédemment, la latence maximale, notée $Lat_M(PO_k)_{Si}$, correspond, pour la période d'observation PO_k, d'un serveur S_i donné, à la durée qui existe entre l'arrivée d'un échec entrant (E↓) sur $S_i(PO_k)$ et le départ du dernier échec sortant (E↑) de $S_i(PO_k)$, qui est lié à cette arrivée. Nous considérons uniquement les échecs sortants (E↑) dans la période PO_k courante.
Ce seront les mécanismes d'alignements coopératifs, entre serveurs, sur les longueurs L_k des période d'observation, que nous verrons plus loin, qui permettront d'intégrer ces échecs sortants (E↑) dans la latence maximale.

Si l'on pose top_p, de PO_k (S_I), comme étant le top_i incluant le dernier échec sortant (E↑) de PO_k dû à un échec entrant (E↓) sur cette période, avec $p\in[1,L_k]$, alors top_p va nous permettre de calculer la latence maximale de PO_k en considérant les trois contraintes suivantes :

'a' : il doit exister au moins un top_j,
avec $j\in[1,p-1]$ tel que $PE_{j\_k}\neq 0$;
'b' : $PE_{p\_k}\neq 0$;
'c' : $\forall top_l$, avec $l\in[p+1,L_k]$, $PE_{l\_k}=0$.

L'idée est que top_p est le dernier top de PO_k dans lequel au moins un échec sortant (E↑) est généré à la suite de l'arrivée du premier échec entrant (E↓) sur PO_k.

Pour définir la probabilité des valeurs (exprimées en longueur via un nombre de top_i) possibles (comprises entre 1 et L_k) de Lat_M, nous devons considérer les containtes distinctes vues ci-avant :

- 'a' : au moins l'un des top_j doit avoir $PE_{j\_k}\neq 0$. Cela s'obtient en considérant 1-Probabilité de ne pas avoir cette propriété pour tous les top_j ;
- 'c' : tous les top_l doivent avoir $PE_{l\_k}=0$. On prendra également 1-Probabilité de ne pas avoir cette propriété. Dans le cas des contraintes 'a' et 'c', les évènements étant indépendants, nous prendrons le produit de ces probabilités. Nous obtenons ainsi la formule générale suivante *(39)* pour le calcul de la latence maximale Lat_M :

$$Prob(Lat_M(PO_k)_{S_i}=p)=\left[1-\prod_{j=1}^{p-1}(1-PE_{j\_k})\right]\times PE_{p\_k}\times\left[\prod_{j=p+1}^{L_k}(1-PE_{j\_k})\right] \qquad (39)$$

IV-B5) Synchronisation des serveurs et période d'observation nominale (PoN_k)

Nous allons approfondir les liens entre les notions de latence minimale ($Lat_m(PO_k)_{Si}$ et maximale $Lat_M(PO_k)_{Si}$), et celle de distance au prochain échec (dPE_{i_j}) afin d'en comprendre les mécanismes profonds.
De cette étude nous esquisserons l'un des résultats observés dans nos simulations et que nous nommerons la période d'observation nominale (PoN). Cette notion est centrale dans notre approche. Elle joue un rôle important dans la synchronisation distribuée des serveurs entre eux en vue d'obtenir un état global du système qui est stable.

Il nous faut revenir sur la notion de propagation d'échec. L'idée est de considérer deux serveurs S_1 et S_2, tous les deux dans un état stable ($a_{1\_k}=a_{2\_p}=0$), et en interaction l'un avec l'autre. Nous nous plaçons sur les périodes d'observation PO_{1_k} de S_1 et PO_{2_j} de S_2.

Dans nos simulations, comme dans la réalité du problème posé, le système fonctionne avec plus de deux serveurs. La simplification, qui consiste à expliquer le phénomène avec deux serveurs, est nécessaire pour mettre en valeur les notions visées.

Les deux serveurs peuvent jouer tantôt le rôle de celui qui est à l'origine d'un échec sortant(E↑), provoquant la création d'une requête résiduelle, ou de celui qui subit un échec entrant (E↓) par l'arrivée d'une requête résiduelle. Ils peuvent aussi jouer simultanément les deux rôles.

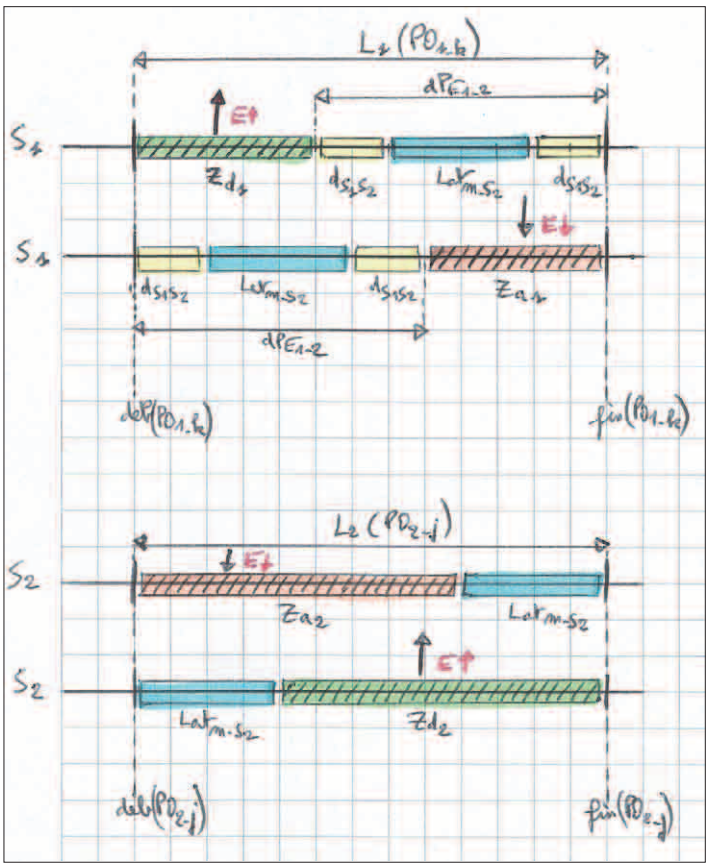


Fig-IV-B5-26

Nous allons regarder ces deux rôles et voir sur quels éléments les deux serveurs S_1 et S_2 peuvent agir pour rester ou retourner le plus vite possible, dans un état stable.
Donnons à S_1 le rôle d'initiateur d'un échec sortant (E↑) et à S_2 celui de récepteur de la requête rédisuelle correspondante (échec entrant (E↓)).
Comme le montre la figure *Fig-IV-B5-26*, nous avons les longueurs des périodes d'observations PO_{1_k} et PO_{2_j} qui sont égales ($L_1=L_2=L$). Ces grandeurs pourraient être différentes et surtout modifiées par les serveurs S_1 et S_2 au cours des heuristiques résolutives.
La distance (en unité logique top_i) entre S_1 et S_2 est notée d_{S1S2}.

La partie haute de la figure *Fig-IV-B5-26* représente le serveur S_1 et ses zones hachurées de départ (Z_{d1} en vert) et d'arrivée (Z_{a1} en orange) d'échecs.
Tout échec sortant (E↑) de S_1 via Z_{d1} peut provoquer en retour de S_2 un échec entrant (E↓) sur S_1 avant la fin de PO_1 via Z_{a1}.
Dans ce cas les échecs peuvent être neutralisés vis-à-vis du processus d'ajustement, laissant le serveur S_1 dans un état stable.

Nous avons :
- Z_{d1} qui est définie par $[deb(PO_{1\_k}), deb(PO_{1\_k})+(L_1-dPE_{1\_2})]$,
- Z_{a1} qui est définie par $[deb(PO_{1\_k})+dPE_{1\_2}, fin(PO_{1\_k})]$, avec les longueurs de Z_{d1} et de Z_{a1} qui sont identiques et valent $(L_1-dPE_{1\_2})$.

La partie basse de la figure *Fig-IV-B5-26* représente le serveur S_2 et ses zones hachurées de départ (Z_{d2} en vert) et d'arrivée (Z_{a2} en orange) d'échecs.

Tout échec entrant (E↓) sur S_2 via Z_{a2} peut générer un échec sortant (E↑) depuis S_2 avant la fin de PO_{2_j} via Z_{d2}.
Dans ce cas les échecs peuvent être neutralisés vis-à-vis du processus d'ajustement, laissant le serveur S_2 dans un état stable.

Nous avons :
- Z_{a2} qui est définie par $[deb(PO_{2\_j}), deb(PO_{2\_j})+(L_2-Lat_m(PO_{2\_j})_{S2})]$,
- Z_{d2} qui est définie par $[deb(PO_{2\_j})+Lat_m(PO_{2\_j})_{S2}, fin(PO_{2\_j})]$, avec les longueurs de Z_{a2} et de Z_{d2} qui sont identiques et valent $(L_2-Lat_m(PO_{2\_j})_{S2})$.

Le début de la période d'observation PO_{1_k} est nommé $deb(PO_{1\_k})$ et sa fin $fin(PO_{1\_k})$.
Nous rappelons que dPE_{i_j} correspond à la distance au prochain échec entrant (E↓), en provenance de S_j, consécutivement à un échec sortant (E↑) de S_i vers S_j.

Prenons le cas où S_1 et S_2 sont stables, respectivement en début de période PO_{1_k} et PO_{2_j} ($a_{1\_k}=a_{2\_j}=0$) et regardons ce qui peut se passer suivant les deux points de vue (S_1 et S_2).

S_1 provoque un échec sortant (E↑) vers S_2 :
- soit un échec sortant (E↑) est généré (avec une probabilité $PE_{\alpha\_k}\neq 0$, pour top_α et PO_{1_k} de S_1) dans Z_{d1}. Nous avons alors deux cas possibles :
  1. (E↓) arrive avant $fin(PO_{1\_k})$. Dans ce cas S_1 reste stable ($a_{1\_(k+1)}=0$). Cela dépend de PE_{β_j} (probabilité pour top_β et PO_{2_j} de S_2), de d_{S1S2} et de $Lat_m(PO_{2\_j})_{S2}$;
  2. aucun échec entrant (E↓) ne revient de S_2 avant $fin(PO_{1\_k})$. Dans ce cas S_1 devient instable ($a_{1\_(k+1)}\neq 0$) pour la période suivante ($PO_{1\_(k+1)}$) ;
- soit un échec sortant (E↑) est généré (avec une probabilité $PE_{\alpha\_k}\neq 0$) dans PO_{1_k}, mais en dehors de Z_{d1}. Dans ce cas l'échec entrant (E↓) arrive (s'il aura été émis, avec une probabilité $PE_{\beta\_j}\neq 0$ par S_2) dans la période suivante PO_{1_k+1} de S_1, rendant ce dernier instable ($a_{1\_(k+1)}\neq 0$). S'il n'arrivait pas du tout (non émis par S_2) on obtiendrait le même résultat.

S_2 reçoit un échec entrant (E↓) depuis S_1. À la réception d'un échec entrant (E↓), généré par S_1, S_2 va émettre ou non, en retour, un échec sortant (E↑) vers S_1. Cela dépendra de la probabilité PE_{β_j}.

Nous avons alors deux cas possibles qui dépendent de la position temporelle dans PO_{2_j} de l'échec entrant (E↓) en provenance de S_1 :

1. (E↓) arrive dans Z_{a2} ; dans ce cas S_2 reste stable ($a_{2\_(j+1)}$=0) puisqu'un échec sortant (E↑) est généré avant la fin de PO_{2_j}. Cela ne dépend que de $Lat_m(PO_{2\_j})_{S2}$, donc de S_2 ;
2. (E↓) arrive dans PO_{2_j}, mais en dehors de Z_{a2} ; dans ce cas $PO_{2\_(j+1)}$ devient instable ($a_{2\_(j+1)}\neq 0$).

QUEL BILAN POUVONS-NOUS TIRER DE CES PROPRIÉTÉS ?

Dans le cas de S_1 (génération d'un échec sortant (E↑)), ce dernier peut jouer sur d_{S1S2} en se déplaçant. Un rapprochement vers S_2 provoquera une diminution de d_{S1S2} et une augmentation de Z_{d1} et inversement en s'éloignant de S_2 cela provoquera une augmentation de d_{S1S2} et une diminution de Z_{d1}. On peut noter que si $d_{S1S2}>L_1$ alors aucun échec sortant (E↑) généré par S_1 ne pourra provoquer d'échec entrant (E↓) sur la même période d'observation, rendant instable S_1. Dans ce cas S_1 aura la possibilité d'augmenter L_1.Nous savons que si $L\to\infty$ alors l'ajustement n'aura plus aucun effet. Au contraire faire tendre 'L' vers des petites valeurs rendra le système beaucoup trop réactif.

Dans le cas de S_2 (récepteur d'un échec entrant (E↓)), ce dernier peut jouer sur Lat_{S2} et sur L_2. Pour augmenter ses chances de rester stable, S_2 peut diminuer Lat_{S2}, qui dépend de PE_{2_j} et/ou augmenter la longueur (L_2) de sa période d'observation ($PO_{2\_(j+1)}$). En effet plus L_2 est importante et plus Lat_{S2} diminue relativement à L_2.

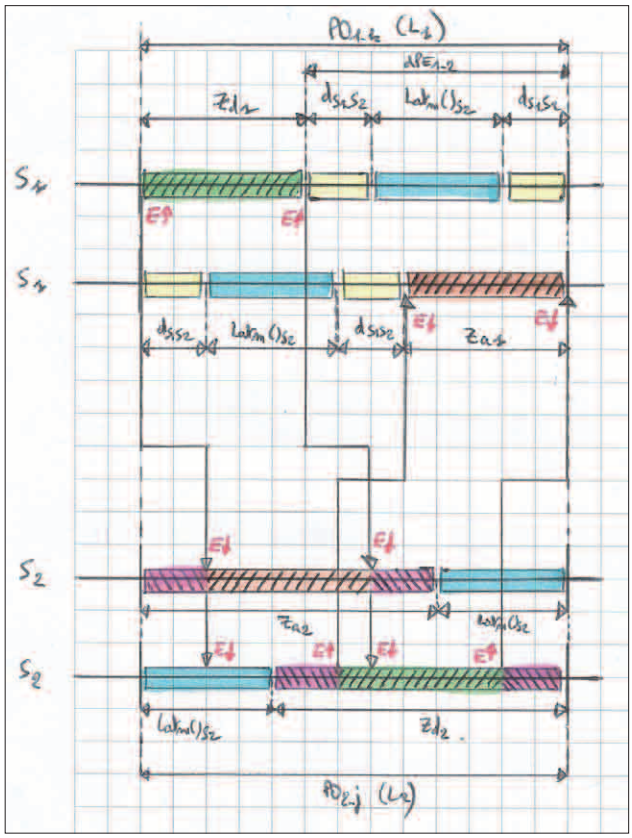


Fig-IV-B5-27

On peut regarder l'effet de la synchronisation ou non des PO entre elles (PO_{1_k} et PO_{2_j}). Sur la figure *Fig-IV-B5-27* les deux PO sont synchronisées sur leur premier top respectif (top_{1_k} démarre en même temps que top_{1_j}). On voit que la distance d_{S1S2} entre les deux serveurs ne joue pas sur les zones de Z_{d1} (en vert) et Z_{a1} (en orange) qui permettent une neutralisation des échecs sortants (E↑).

En revanche cette distance impacte directement S_2 en réduisant de deux fois cette distance, les parties (orange et verte) de Z_{d2} et Z_{a2} qui permettent de neutraliser les échecs entrants (E↓).

Plus la distance d_{S1S2} est grande et moins la valeur effective des zones Z_{d2} et Z_{a2} est grande. Si $d_{S1S2} \geq Z_{d2}$ ou Z_{a2} alors la valeur effective de Z_{d2} et de Z_{a2} vaudra 0.
À l'inverse, si d_{S1S2} est très petite ($\rightarrow 0$) alors Z_{d2} et Z_{a2} seront pleinement utilisées (pas de zone rouge).
On en déduit la règle suivante.

Règle: la distance (d_{S1S2}) entre deux serveurs (S_1 et S_2) qui interagissent (de S_1 vers S_2) doit être strictement inférieure à la longueur des zones de départ (Z_{d2}) des échecs sortants (E↑) et d'arrivée (Z_{a2}) d'échecs entrants (E↓) sur S_2.

Nous venons de voir que les serveurs n'ont pas nécessairement intérêt à être synchronisés, surtout s'ils sont éloignés l'un de l'autre.

Regardons ce qui se passe si les serveurs ne sont pas synchronisés.

Sur la figure *Fig-IV-B5-28* nous avons décalé les périodes PO_{1_k} et PO_{2_j} de la distance entre S_1 et S_2 (d_{S1S2}) afin de resynchroniser le début des périodes (de longueurs identiques) sur leur premier top.
On s'aperçoit que cela ne corrige pas le problème soulevé sur la figure *Fig-IV-B5-27*, lorsque les serveurs étaient synchronisés.

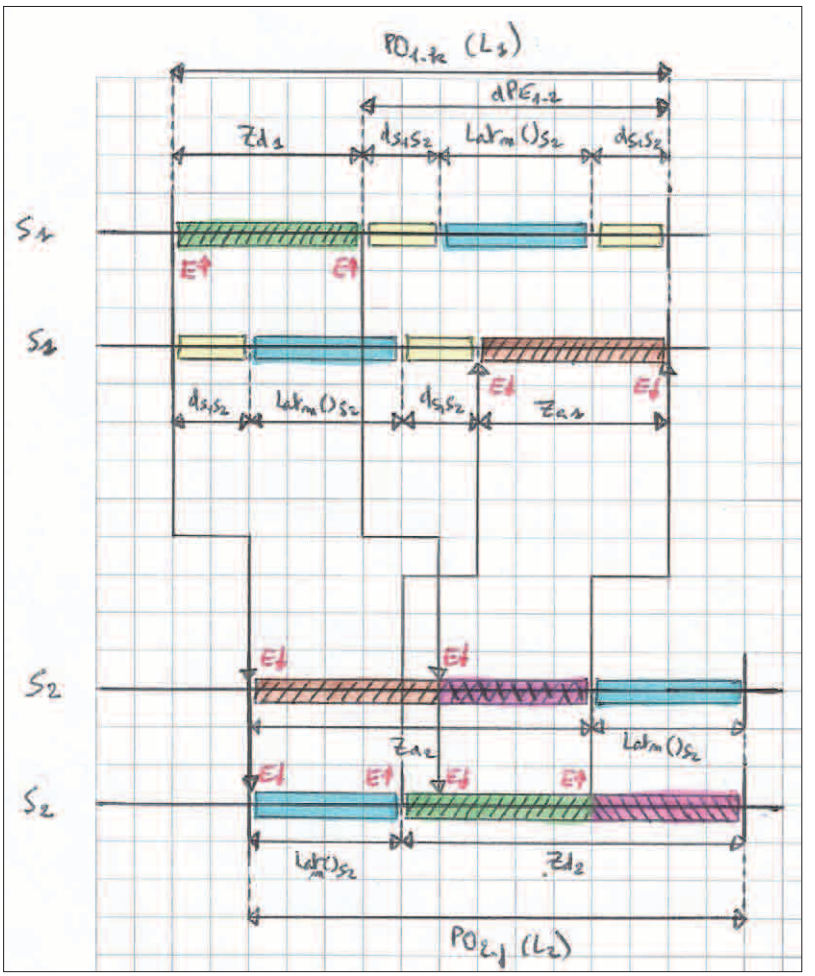


Fig-IV-B5-28

Définition: La fonction Lg(Z) correspond à la longueur de la zone 'Z'.

Pour résoudre cette difficulté il nous faut formaliser le problème qui consiste à résoudre l'équation:

$$Lg(Z_{d1}) = Lg(Z_{a2}), \qquad (40)$$

avec $Lg(Z_{d1}) = Lg(Z_{a1})$ et
$Lg(Z_{d2}) = Lg(Z_{a2})$.

Pour résoudre *(40)* il faut partir des éléments suivants:

$$Lg(Z_{d1}) + 2d_{S1S2} + Lat_m(PO_{2\_j})_{S2} = L_1,$$
$$LG(Z_{a2}) + Lat_m(PO_{2\_j})_{S2} = L_2,$$
$$\Rightarrow L_1 - 2d_{S1S2} - Lat_m(PO_{2\_j})_{S2} = L_2 - Lat_m(PO_{2\_j})_{S2}$$
$$\Leftrightarrow (L_1 - L_2) = 2d_{S1S2}.$$

Nous obtenons dans le cas général:

$$d_{S1S2} = (L_1 - L_2)/2. \qquad (41)$$

Qui peut aussi être écrite avec L_1 en fonction de L_2 et d_{S1S2}:

$$L_1 = 2d_{S1S2} + L_2. \qquad (42)$$

Un cas particulier de *(41)* est celui où $L_1=L_2 \Rightarrow d_{S1S2}=0$.

Pour résoudre *(40)* deux solutions complémentaires se présentent pour vérifier *(41)* :
- soit en faisant varier la distance (d_{S1S2}) entre les serveurs ;
- soit en faisant varier L_1 et/ou L_2.

Si l'on a $L_1=L_2=13$ et $d_{S1S2}=2$, nous avons vu que *(41)* n'était pas vérifié.
Pour régler le problème dans ce cas nous pouvons changer d_{S1S2}, ou L_1 et/ou L_2.

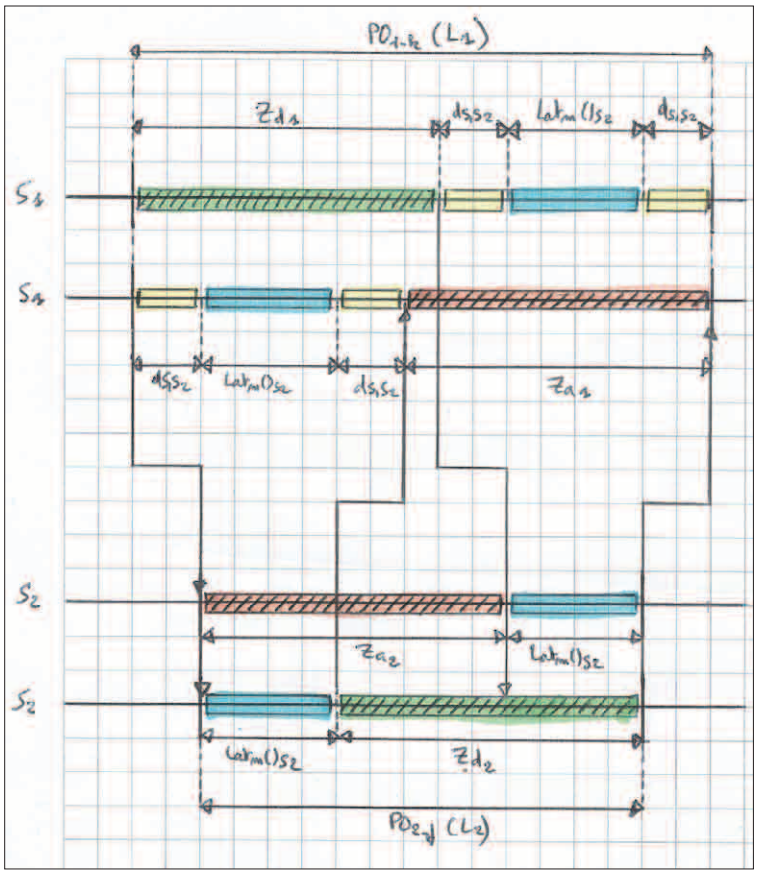


Fig-IV-B5-29

Sur la figure *Fig-IV-B5-29* en partant de $L_1=17$ et $d_{S1S2}=2$, nous avons choisi de changer L_2 afin que *(41)* soit vérifiée. Nous obtenons donc $L_2=13$.
Le problème est bien résolu, c.-à-d. les zones Z_{d1}, Z_{a1}, Z_{d2} et Z_{a2} sont bien utilisées entièrement, mais à la condition d'avoir aussi désynchronisé S_1 et S_2 d'une longueur égale à d_{S1S2}.

En réalité cette désynchronisation d'une longueur égale à la distance entre les serveurs, n'est pas suffisante. Cela fonctionne pour la première occurrence des deux périodes d'observation des serveurs. Dans la mesure où les longueurs L_1 et L_2 sont différentes, les écarts entre les occurrences vont changer.

Un autre problème apparaît avec *(41)* dans la mesure où dans une interaction entre plusieurs serveurs, chacun d'entre eux peut jouer alternativement le rôle de générateur d'échecs sortants (E↑) et de récepteur d'échecs entrants (E↓). Dans ce cas il faudrait pouvoir vérifier *(41)* de $S_1 \rightarrow S_2$ et réciproquement.
Cela donnerait simultanément :
$L_1=2d_{S1S2}+L_2$,
$L_2=2d_{S1S2}+L_1$.

Ceci n'est possible que si $d_{S1S2}=0$, ce qui n'est pas réaliste.

Pour comprendre ces phénomènes nous allons faire des frises (cf. les figures *Fig-IV-B5-30/31*) dans lesquelles deux serveurs S_1 et S_2 interagissent. S_1 a une longueur de PO_{1_k}, $L_1=5$, alors que S_2 a une longueur de PO_{2_j}, $L_2=3$. Pour compléter le dispositif, nous aurons $Lat_m(PO_{2\_j})_{S2}=1$ et $d_{S1S2}=1$.

Définition : On appelle décalage temporel entre deux occurrences de périodes d'observation (PO_{1_k} et PO_{2_j}) appartenant à des serveurs (S_1 et S_2) en interaction, la grandeur δ_{k_j} qui vérifie : $\delta_{k\_j}=|deb(PO_{2\_j})-deb(PO_{1\_k})|$.

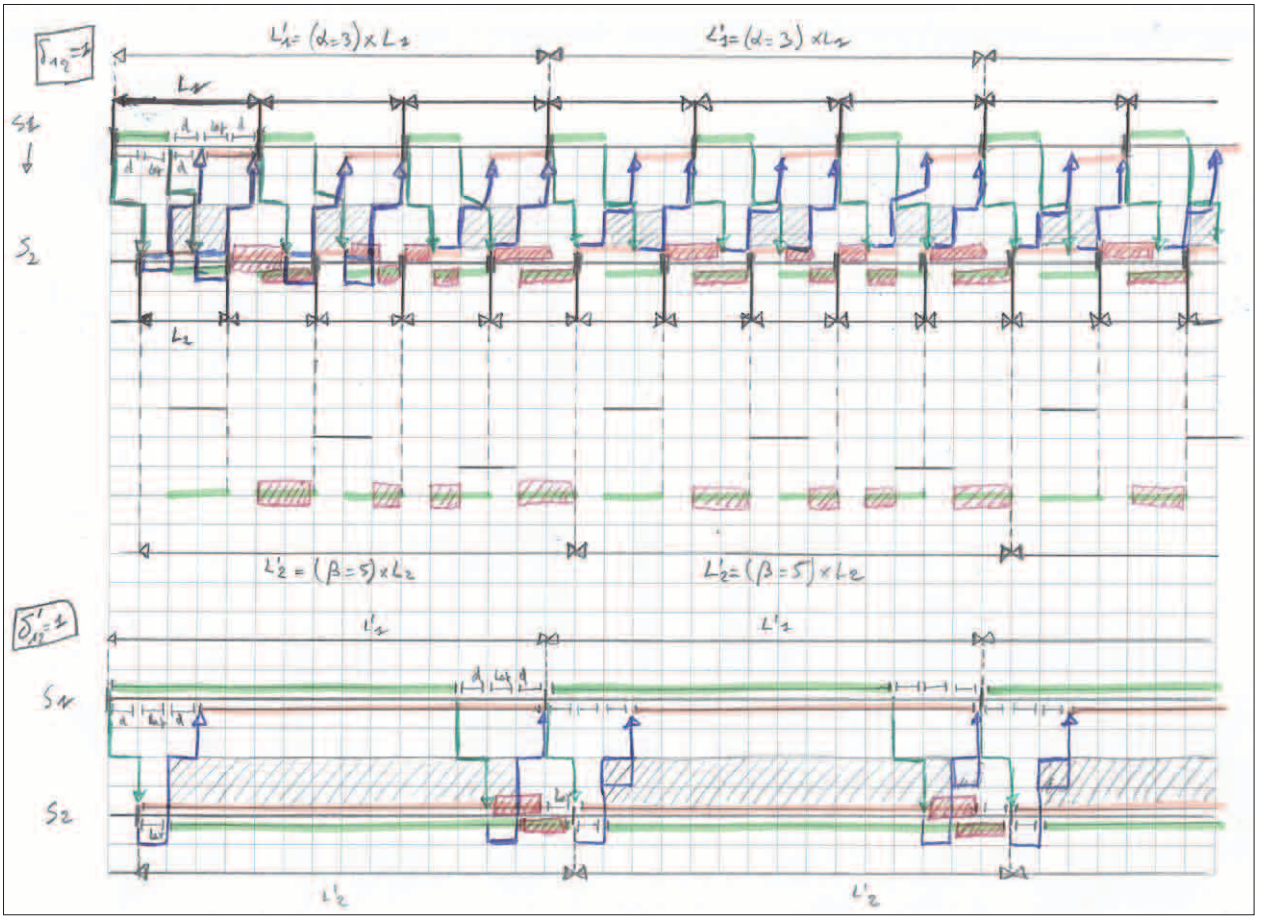


Fig-IV-B5-30

Nous allons étudier trois configurations. Elles auront respectivement $\delta_{k\_j}=d_{S1S2}$, $\delta_{k\_j}=2d_{S1S2}$ et $\delta_{k\_j}=0$.
Sur la figure *Fig-IV-B5-30*, $\delta_{k\_j}=d_{S1S2}$. Cela signifie qu'au début de la frise (partie haute) PO_{2_1} démarre avec un décalage $d_{S1S2}=1$ avec PO_{1_1}.
En vert nous avons représenté les zones Z_{d1} et Z_{d2}, alors que les zones Z_{a1} et Z_{a2} sont représentées en orange. Les flèches en vert représentent des échecs potentiels sortants (E↑) de S_1 vers S_2, alors que les flèches en bleu les échecs potentiels sortants (E↑) de S_2 vers S_1, consécutivement aux arrivées d'échecs en provenance de S_1.

On peut observer sur la frise, et plus particulièrement sur la partie centrale de la figure *Fig-IV-B5-30*, que le décalage est cyclique. En effet au bout de trois PO_{1_k} et cinq PO_{2_j} on retrouve la situation initiale, à savoir $\delta_{k\_j}=d_{S1S2}$.
Ce cycle répond à l'équation *(43)* suivante :

$$\alpha L_1=\beta L_2. \qquad (43)$$

En effet pour retrouver la même situation il faut qu'un multiple de L_1 soit égal à un autre multiple de L_2. Cela donne :

$$\alpha/\beta=L_2/L_1. \qquad (44)$$

Dans notre exemple $L_1=5$ et $L_2=3$. Le cycle opère bien si $\alpha=3$ et $\beta=5$, mais aussi si $\alpha=6$ et $\beta=10$,…

On observe également, sur la figure *Fig-IV-B5-30*, que le décalage ne joue pas directement sur l'efficacité optimale de S_1, puisque les zones Z_{d1} (en vert) et Z_{a1} (en orange) sont utilisées en totalité. En revanche ça n'est pas le cas pour S_2, puisque de nombreuses zones en rouge

hachuré montrent que les zones Z_{d2} et Z_{a2} sont loin d'être entièrement utilisées, en dehors de la synchronisation (une fois par cycle) avec $\delta_{k\_j}=d_{S1S2}$.
Cela signifie que la gestion de S_1 est efficace dans l'interaction avec S_2, mais la réciproque ne l'est pas.
Or si S_2 est instable, il génèrera plus d'échecs sortants (E↑) vers S_1, qui le déstabiliseront à son tour. Dans l'intérêt de l'interaction, il faut trouver une solution qui satisfasse les protagonistes. D'où l'idée d'augmenter la taille des périodes d'observation afin de synchroniser les serveurs.
L'idée la plus simple est de se positionner sur le plus petit multiple commun. C'est ce que nous avons fait dans la partie basse de la figure *Fig-IV-B5-30*, dans laquelle L'_1 devient la nouvelle longueur de PO_{1_k} et L'_2 celle de PO_{2_j} telles que $L'_1=\alpha L_1$ et $L'_2=\beta L_2$, avec $\alpha=3$ et $\beta=5$.
La situation s'est grandement améliorée puisque les zones en rouge hachuré, n'apparaissent, pour S_2, que marginalement en fin de Z_{a2} et Z_{d2}, avec une longueur de deux unités logiques (top_i). Les zones Z_{d1}, Z_{a1}, Z_{d2} et Z_{a2} sont nettement mieux exploitées qu'avec les longueurs initiales L_1 et L_2. Cela aura pour conséquence de diminuer très sensiblement la probabilité de générer des échecs sortants (E↑).

plus petit multiple commun

Propriété: La co-construction décentralisée (via l'interaction entre S_1 et S_2) des longueurs L'_1 et L'_2 des périodes d'observation PO_{1_k} et PO_{2_j}, basée sur le plus petit multiple commun, synchronise les serveurs et accélère leur stabilité.

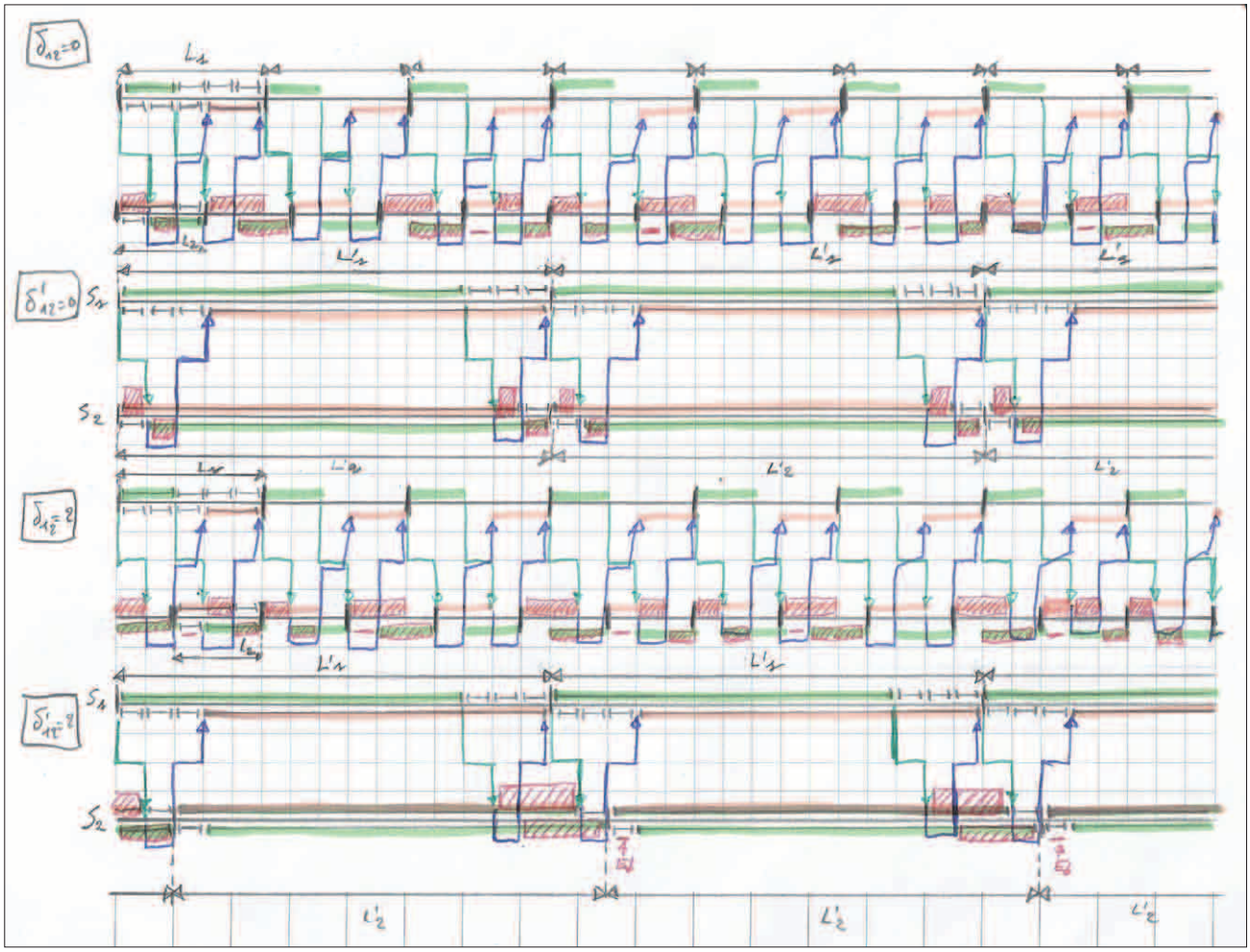


Fig-IV-B5-31

Pour compléter cette étude nous allons prendre sur la figure *Fig-IV-B5-31* deux autres situations, où $\delta_{k\_j}=0$, sur la partie haute, et $\delta_{k\_j}=2d_{S1S2}$ sur la partie basse. Cela confirme les résultats obtenus sur la figure *Fig-IV-B5-31*. Les zones en rouge hachuré sont nombreuses dans les deux cas pour L_1 et L_2, alors qu'elles sont réduites en début et en fin de PO_{2_j} pour S_2 avec $\delta_{k\_j}=0$ et en fin de PO toujours pour S_2 avec $\delta_{k\_j}=2d_{S1S2}$. On peut noter que pour un décalage d'une unité on a une perte de deux unités en fin de Z_{d2} et Z_{a2}, et pour un décalage de deux unités on a une perte de trois unités en fin de Z_{d2} et Z_{a2}. Il est simple de voir que cette perte croît d'une unité avec l'augmentation du décalage d'une unité.

Nous venons de voir que lorsque deux serveurs, ou plus, interagissent, ils ont intérêt à ne pas prendre une PO de taille trop petite, et à se synchroniser sur une taille commune pour leur PO basée par exemple sur le plus petit multiple commun.
La période d'observation d'un serveur correspond à une durée sur laquelle ce dernier répartit de façon homogène ses efforts de production, pour mieux répondre aux demandes qui lui seront faites sur cette même durée à venir.
Nous avons vu qu'il existe une probabilité PE_{i_k} pour que le serveur 'S' génère des échecs sortants (E↑) pendant top_i de PO_k.
D'après *(20)*, l'ajustement a_{k+1} dépend de $(1/L_k)$. En allongeant L_k, on augmente le nombre de top_i (unité temporelle logique) et on diminue la production p_k associée. Il est simple de voir que cette diminution augmente la probabilité PE_{i_k}, lors d'arrivées de demandes qui dépassent cette valeur p_k. Bien que relativement marginal cela va dans le sens de ne pas avoir des PO avec des longueurs trop grandes.

Cette remarque met en évidence le problème, non pas de demandes qui ne seraient pas constantes dans le temps, mais de celui de la distribution des demandes dans le temps.
Si les demandes arrivent de façon constante sur L_k, alors l'ajustement corrige la production répartie sur chaque top et les échecs dûs à PE_{i_k} ne le seront qu'à cause de l'ordre du choix des évènements (demandes, production, requêtes résiduelles). En revanche si la distribution de l'arrivée des demandes sur L_k n'est pas du tout homogène, les choses se compliquent.

Comme le montre la figure *Fig-IV-B5-32*, il pourrait y avoir de nombreuses demandes classiques arrivant sur une ou plusieurs parties de PO_{i_k}, alors que d'autres pourraient ne pas en avoir.

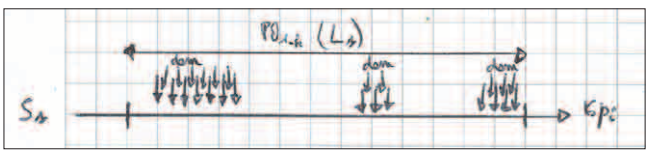


Fig-IV-B5-32

Cette hétérogénéité de l'arrivée des demandes conduit à des PE_{i_k} très disparates. On pourrait alors soit chercher à réduire L_k pour mieux encadrer les zones où les arrivées sont homogènes, soit au contraire chercher à augmenter L_k pour se donner du temps pour amortir les inévitables productions d'échecs sortants (E↑).

La première approche est en contradiction avec le principe de l'ajustement qui nécessite des L_k constantes.
La distribution des demandes apparaît comme une contrainte supplémentaire pour trouver la bonne valeur de L_k qui permettra de gérer au mieux (neutraliser) les échecs sortants (E↑).

PoN_k

Définition: Nous appelons période d'observation nominale (PoN_k) du serveur S_k, la période d'observation de S_k dont la longueur est optimale pour gérer la convergence de son processus d'ajustement dans les contraintes qu'il rencontre.

On trouve dans ces contraintes sa propre génération d'échecs sortants (E↑), liés à la distribution des demandes et à l'ordre de traitement des évènements sur chaque top_i.

La PoN_k est donc propre à un serveur S_k indépendamment de tout processus de coopération avec d'autres serveurs.

Comme nous l'avons évoqué juste avant, une synchronisation entre serveurs nécessite de faire évoluer les longueurs des PO de chacun des serveurs, vers une valeur commune calculée, par exemple, sur la base du plus petit multiple commun.

Principe: L'idée, que nous expérimenterons dans nos simulations, est que chaque serveur doit trouver sa PoN pour optimiser son propre fonctionnement. Dès qu'un serveur devra recourir à la coopération avec d'autres serveurs pour neutraliser des échecs entrants (E↓), soit exogènes (provenant d'autres serveurs), soit endogènes (provenant de lui-même, consécutivement à des requêtes résiduelles), il fera varier temporairement la longueur de sa PO afin de se synchroniser avec le serveur visé. Une fois la stabilisation atteinte, il pourra reprendre la taille initiale de sa PoN.

Le calcul de la longueur L_k de PoN, pour un serveur donné, est donc le résultat d'un processus d'adaptation de ce serveur vis-à-vis de son environnement (volume et distribution des demandes). En mesurant les échecs sortants (E↑) il peut adapter la longueur L_k de sa période d'observation jusqu'à obtenir une valeur qui lui permet de neutraliser ces échecs. Il aura ainsi trouvé seul sa PoN.

Les hypothèses de nos expérimentations se basent sur des demandes constantes de la part des clients. En réalité ces demandes pourraient évoluer, conduisant les serveurs à remettre en question la valeur de leur PoN.
Si les demandes n'avaient aucune stabilité/constance les serveurs ne pourraient pas se stabiliser.
L'avantage de l'organisation en serveurs mobiles est de pouvoir travailler sur des demandes en moyenne, par région/secteur.
Cela autorise des variations locales beaucoup plus réactives que celles en moyenne sur lesquelles le système s'ajustera.

1 « Suffixe latin qui signifie celui qui agit ; *cantoris* est celui qui chante / le chanteur », exemple tiré de la thèse de Fabrice Harrouet intitulée « *oRis : s'immerger par le langage pour le prototypage d'univers virtuels à base d'entités autonomes* », Enib, octobre 2000.
factoris est le créateur, l'inventeur, l'auteur, ou encore le fabricant, celui qui fabrique, *pictoris/coloratoris* est le peintre, celui qui peint…

Pour donner corps à nos heuristiques nous avons très vite choisi d'utiliser l'outil *oRis* proposé et développé par Fabrice Harrouet à l'École Nationale d'Ingénieurs de Brest dans le laboratoire d'informatique Industriel (LI2).
Cet outil extrêmement intéressant, créé dans le contexte d'un atelier de réalité virtuelle (*Arévi*) à donné lieu à un mémoire de thèse soutenu par l'auteur en novembre 2000 et a été repris dans des travaux couvrant un spectre de recherches très varié. On peut citer celui de l'informatique théorique, par le biais d'une thèse intitulée «*Premiers éléments de la Théorie du Calcul Singulier*» soutenue en juin 2002 par Roger Cozien, celui très appliqué en biologie autour par exemple d'une modélisation de la coagulation sanguine dans les veines, ou encore celui du partage de ressources dans les réseaux informatiques.

IV-C1) Rapports de Doctorat et d'Habilitation à Diriger des Recherches

Rapport sur les travaux présentés par Fabrice Harrouet pour obtenir le titre de docteur de l'Université de Bretagne Occidentale, en Informatique, intitulé :
«*oRis : s'immerger par le langage pour le prototypage d'univers virtuels à base d'entités autonomes*».

Caen, le 22 novembre 2000.

Le sujet abordé par Fabrice Harrouet se situe dans le domaine de la «réalité virtuelle» et plus particulièrement dans celui de l'utilisation pragmatique des techniques de «réalité virtuelle» pour le prototypage interactif. L'auteur aborde cette question en proposant un outil dont les objectifs sont d'observer, d'expérimenter et de façonner des modèles numériques (artificiels), comme nous pouvons le faire sur des systèmes réels. L'accent est mis sur la représentation du comportement des composants de ces modèles numériques, dans lequel peut intervenir l'apparence de ces composants (aspects de rendu en deux ou trois dimensions propres à la réalité virtuelle) comme un ensemble de paramètres parmi d'autres. L'auteur défend l'idée d'un «prototypage interactif» par un procédé qu'il nomme «immersion langagière» en référence à l'immersion multi-sensorielle en «réalité virtuelle». Grâce à ce procédé fondé sur une approche langagière (au sens informatique du terme), qui repose sur le changement dynamique (en cours de fonctionnement) du comportement des composants impliqués dans les modèles numériques manipulés, il est donc possible de compléter les modèles existants par l'introduction de nouveaux modèles, ou de modifier les modèles en cours, afin de mieux comprendre leurs impacts sur le système étudié.
Cela le conduit à rapprocher ses préoccupations de celles défendues dans la communauté des Systèmes Multi-Agents, sur la question du fonctionnement d'entités dites «autonomes» dans un environnement situé. Ce double héritage (SMA et réalité virtuelle) enrichit sa réflexion pour produire un cadre expérimental utilisable dans ces deux domaines,

comme l'attestent les différentes expérimentations faites avec l'outil proposé.

Le document se compose de quatre parties. La première partie pose à la fois les exigences attendues par l'auteur pour son étude (au chapitre 2) et fait le point sur les principaux outils et mécanismes aux centre de la question posée (autonomie au chapitre 3 et comportement dynamique au chapitre 4). Dans la deuxième partie Fabrice Harrouet présente de façon détaillée (les fonctionnalités au chapitre 5 et des applications significatives au chapitre 6) l'outil *oRis* qui est le résultat de ses contributions. La troisième partie (au chapitre 7) donne l'occasion à l'auteur de faire un bilan sur son travail et de livrer ses réflexions sur une suite à donner dans cette direction. La quatrième et dernière partie se compose d'une suite d'annexes qui développent les applications présentées au chapitre 6, et apportent un complément d'informations sur certaines propriétés d'*oRis* proposées sous la forme de paquetages standards.

Dans le chapitre 2 l'auteur pose les bases de ce qu'il défend dans la suite de son travail, autour de la notion de «simulation interactive» ou encore appelé «prototypage interactif» dans laquelle les modèles numériques du monde réel doivent être accessibles en cours de fonctionnement pour être affinés en fonction des perceptions et des expérimentations faites. Ceci implique la réalisation d'univers virtuels s'exécutant de façon ininterrompue quelles que soient les interventions des expérimentateurs. La plasticité de l'outil informatique, dont l'auteur donne un éclairage tout-à-fait pertinent dans les chapitres 3 et 4, mais aussi dans les annexes (chargement dynamique de code, ordonnancement et parallélisation), va lui servir de base technique pour la concrétisation de ses idées.
Le chapitre 3 s'intéresse aux procédés d'activation des entités autonomes. Après une introduction et un plaidoyer pour l'approche individu-centré qui consiste à décrire une population à travers les comportements des individus qui la composent, l'auteur s'interroge sur la manière (technique) de la mettre en œuvre. L'auteur mène une réflexion intéressante sur les notions (et travaux s'y rapportant) d'objets, d'objets actifs, d'agents et de systèmes multi-agents. Il souligne à cette occasion le rôle primordial du procédé d'activation des entités, qui doit assurer l'équité des traitements. Le procédé d'ordonnancement des différents comportements élémentaires ne doit donner lieu à aucun biais qui fausserait la valeur des résultats observés.
Fabrice Harrouet étudie au chapitre 4 les propriétés offertes par les différents langages de programmation en matière de comportement dynamique (modification à l'exécution). Son étude montre qu'il existe plusieurs types de comportements dynamiques, qui ne sont généralement pas tous présents dans un même langage de programmation. On trouve le typage dynamique des données, le chargement dynamique de modules ou encore l'introduction de code source en cours d'exécution. Ce constat l'amène à proposer au chapitre 5 son propre langage *oRis* qui satisfait les besoins exprimés au chapitre précédent.

Le chapitre 5 est la contribution de l'auteur à ce qu'il appelle l'immersion langagière. Il s'agit de proposer un langage informatique qui étend les propriétés des langages classiques (en particulier *C++* et *Java*) de telle façon que tous les critères retenus dans son analyse sur le prototypage interactif soient présents. De nombreux mécanismes sont introduits comme par exemple celui du contrôle d'accès dynamique, qui permet de contrôler par l'objet invoqué à l'exécution quel est l'objet qui l'invoque. Ce mécanisme est possible car le langage *oRis* peut inspecter la pile d'exécution du traitement en cours. Cette fonctionnalité va d'ailleurs dans le sens du paradigme «agent» et de la notion d'entité autonome, qui doit contrôler d'où proviennent les interactions pour ne pas les subir. Parmi les mécanismes les plus importants on trouve celui de l'introspection simple pour faciliter l'interaction utilisateur-application, tout ce qui touche au parallélisme (contrôle d'ordonnancement des comportements d'entités autonomes), les différents modes de communication (synchrone, lien réflexe, message point-à-point, diffusion de messages), la notion d'environnement (entités situées) et enfin les aspects dynamiques (introduction de code, déclenchement de traitements, intervention sur les fonctions, classes ou instances).

Fabrice Harrouet illustre dans le chapitre 6 l'utilisation du langage *oRis* avec plusieurs exemples qui mettent en valeur ses propriétés (ordonnancement et diffusion de messages) mais aussi et surtout l'approche du «prototypage interactif» par immersion langagière. Les exemples, où l'on pilote la rotation automatique de la poignée d'un pignon en fonction d'une aiguille de l'horloge, ou à l'inverse l'on fait pivoter l'horloge selon les mouvements du pignon choisi, illustrent très bien le principe d'interconnexion des applications pré-existantes l'une à l'autre, fondé sur des modifications de comportements très localisées.
Dans le chapitre 7, Fabrice Harrouet présente ses conclusions et perspectives en cours ou à donner à son travail. Cela lui donne l'occasion de montrer que son travail n'a pas été réalisé de façon isolée, mais s'inscrit dans un projet ambitieux la plate-forme *AréVi* pour la réalité virtuelle, dont *oRis* occupe une place centrale.

Enfin, dans les annexes, Fabrice Harrouet détaille (qualitativement et quantitativement) les résultats de tests qu'il a effectué sur la façon dont sont implantées les propriétés liées au parallélisme (ordonnancement des "threads") et à l'extension dynamique dans les langages classiques (*Java*, *Smalltalk*, *C++*,…). La mise en œuvre de ces tests lui ont permis d'acquérir les compétences nécessaires pour la réalisation de ces propriétés dans *oRis*, conformément aux objectifs qu'il s'était fixés initialement. On trouve également dans ces annexes, une description des principaux paquetages fournis avec le langage et qui l'enrichissent de mécanismes supplémentaires.

Au-delà de l'outil *oRis* et de son langage associé, proposés par Fabrice Harrouet, sa principale contribution me semble être de montrer la faisabilité et l'intérêt décisif d'une démarche d'expérimentation virtuelle de modèles pour la

compréhension de phénomènes complexes étudiés. Cet outil, combinant les avancées en matière de langage informatique, de conception multi-agents et de réalité virtuelle, ouvre la voie à d'autres outils et réflexions qui permettront aux chercheurs de conceptualiser leurs objets d'étude de façon beaucoup plus expérimentale que par le passé. Ce point est fondamental pour aborder tout système complexe (réel) en particulier dans la recherche d'hypothèses sur leur fonctionnement. En effet, dans tous les systèmes complexes étudiés, il apparaît ce que l'on pourrait appeler des phénomènes intermédiaires (qui ne sont observables ni en début, ni en fin d'expérience, si tant est que ces notions de début et de fin aient un sens!), qui à eux seuls expliquent des parties décisives dans le comportement global de ces systèmes. Ces phénomènes cachés conduisent le plus souvent à faire des interprétations partielles qui sont traduites dans des modèles non seulement incomplets, mais le plus souvent inexacts. Ceci est du au fait qu'il est impossible de mathématiser les comportements de ces systèmes (cf. les problèmes de conditions initiales dans les systèmes chaotiques) et donc de prévoir justement ces parties «obscures». Ce n'est pas un hasard si les quelques domaines d'applications évoqués par l'auteur dans le champ d'expérimentation de l'outil, sont apparemment très différents les uns des autres. En réalité ils abordent tous les mêmes conditions d'étude, à savoir les systèmes complexes.

Devant les difficultés auxquelles nous sommes confrontés pour proposer des hypothèses pertinentes concernant par exemple les phénomènes intermédiaires, la voie proposée par Fabrice Harrouet me semble très prometteuse. En effet aujourd'hui on ne voit pas comment aborder des comportements aussi complexes (imprévisibles et cachés) sans tenter de les reproduire le «plus près» possible de l'endroit où l'on suppose qu'ils interviennent. Ceci nous oblige à tester «à la volée» des modèles comportementaux locaux et à réagir (observer les liens de causalité, comprendre les influences mutuelles ou encore aborder le rôle de mécanismes de mémorisation/oubli dans les comportements des composants du système) en conséquence.

Mais là où l'auteur apporte un début de réponse (ouvre une brèche), intervient très vite des questions délicates, au sujet de la description et de la manipulation de modèles comportementaux. En effet, l'exemple du sablier me semble significatif d'une certaine simplification du problème, même si elle peut être recherchée à des fins pédagogiques. Dans cet exemple le modèle correspond à la modification d'une formule physique (signe + ou -), or dans les systèmes multi-agents en général, le modèle comportemental est beaucoup moins évident à isoler. Il faut reconnaître que dans l'approche individu-centrée, justement les comportements individuels sont volontairement simples et donc plus facilement explicitables. Cela n'empêche qu'il faut accompagner cette approche d'une réflexion approfondie sur ce que l'on entend par modèle comportemental et représentation de ces modèles.
Il me faut toutefois revenir sur l'outil *oRis* (je prends l'outil au sens large, c.-à-d. incluant le langage) lui-même,

dans la mesure où il est le garant d'une démarche réaliste et donc prometteuse. À ce sujet il faut souligner le très grand soin avec lequel Fabrice Harrouet à conçu et mis au point cet outil. Je regrette d'ailleurs qu'une comparaison directe (et non au travers des langages de programmations utilisés) de son outils avec ses «concurrents», en terme fonctionnel bien sûr, mais aussi et surtout en terme de propriétés de fonctionnement (efficacité, équité,…), n'ait été faite. Elle montrerait justement en quoi la contribution de l'auteur, eu égard à la façon dont on conçoit et réalise de tels outils, à pris une place déterminante dans le résultat, non pas seulement en terme de commodité d'emploi (travers de beaucoup d'outils actuels), mais bien surtout en terme de pertinence des résultats obtenus.

Il m'apparaît également que son travail pose des questions importantes en terme de réalisation informatique, que l'auteur aurait pu davantage développer; cela n'apparaît que dans la partie perspective. En particulier on voit que les choix de conception ont portés sur l'introduction dans le noyau de l'outil, de mécanismes profonds (par exemple l'algorithme de l'exécution équitable des agents dans le fonctionnement des simulations), non nécessairement visibles, mais en tous les cas indispensables, et de proposer une ouverture avec des paquetages (comme par exemple la diffusion de messages – bien que sur ce point je reste persuadé que l'introduction dans le noyau d'un modèle synchrone est préférable, comme le souligne d'ailleurs l'auteur dans ses perspectives – ou encore la connexion TCP/IP sur les réseaux). Cela conduit à un outils très performant et somme tout assez léger d'installation et d'utilisation. Cette forme de modularité, touchant en plus aux applications de type multi-agents, pourrait conduire au moins à deux extensions qui me paraissent intéressantes. Ici on touche à l'ambiguïté du statut de l'outil proposé. S'agit-il d'un simulateur ou doit-il évoluer vers un outil utilisable pour le propre fonctionnement du système étudié. En effet on peut considérer l'outil comme un élément architectural important, comme le souligne l'auteur dans ses perspectives lorsqu'il parle de la notion de prototypage interactif et coopératif. Mais on peut aussi le voir comme un moteur parmi d'autres inclus dans des entités autonomes (agents) pour justement construire des modèles comportementaux sur le principe du «test&try» en direct. Il me semble que cette voie est assez prometteuse, car elle va dans le sens de la recherche d'autonomie pour les artefacts informatiques, capable d'explorer les rouages les plus intimes des systèmes complexes. Enfin une troisième voie est celle que l'outil occupe aujourd'hui dans son rôle d'aide à la réflexion pour la compréhension de phénomènes complexes.

Enfin il faut souligner la clarté du manuscrit qui est très agréable à lire et sa construction qui reflète étroitement la démarche méthodologique adoptée par l'auteur.
Ce travail très abouti et plein de perspectives comme le prototypage réparti et coopératif, s'inscrit dans un projet de grande ampleur (*AreVi*), auquel Fabrice Harrouet a apporté très largement une contribution plus que significative. Cela dénote le sérieux de l'auteur

et une certaine forme de courage, dans la mesure où la réalisation d'outils opérationnels comme *oRis*, n'est pas toujours bien perçu par la communauté académique, dans la mesure où il est assez facile aujourd'hui (avec les outils de développement) de faire des logiciels qui n'apportent pas grand chose. Fabrice Harrouet a relevé ce défi en fournissant à la communauté scientifique un outil de très haute technicité, directement opérationnel (déjà utilisé dans différents projets) et pertinent pour reconsidérer l'approche des systèmes complexes.

Pour toutes ces raisons et la qualité du travail effectué par Fabrice Harrouet je suis favorable à ce qu'il présente son travail en vue de l'obtention éventuelle du grade de Docteur de l'Université de Bretagne Occidentale.

François Bourdon
Professeur à l'Université de Caen.

Rapport sur les travaux présentés par Jacques Tisseau pour obtenir l'Habilitation à Diriger des Recherches (HDR) de l'Université de Bretagne Occidentale, en Informatique, intitulé :
«*Réalité Virtuelle - autonomie in virtuo* -».
Caen, le 20 novembre 2001.

Jacques Tisseau présente dans un document de synthèse intitulé «*Réalité Virtuelle –autonomie in virtuo–*», les travaux de recherche en informatique qu'il a mené depuis son recrutement comme Maître de Conférences 61e section en 1988 à l'École Nationale d'Ingénieurs de Brest (ENIB). Ces travaux novateurs dans le domaine de la réalité virtuelle, ont contribué à montrer en quoi cette nouvelle discipline des sciences de l'ingénieur ne se limitait pas à des techniques de rendu graphique ou à des dispositifs multisensoriels. Ils posent clairement les questions suivantes : *quels concepts ?, quels modèles ? et quels outils ?* pour la réalité virtuelle, avec comme concept clé *l'autonomie*. Le document de synthèse est articulé en trois parties qui correspondent aux questions posées.

La première partie (chapitre 2) retrace la genèse du concept de réalité virtuelle depuis son apparition dans les années 1990 ; elle conduit l'auteur à postuler un principe d'autonomie, au travers duquel le *virtuel* devient une construction dans l'univers du modèle sensé représenter le réel. L'auteur parle donc d'univers virtuels réalistes et participatifs, comme étant composés d'un ensemble de modèles numériques autonomes en interaction les uns avec les autres. Les modèles numériques sont ainsi mis au centre du concept de réalité virtuelle. L'autonomie dont disposent ces modèles leur permet, via des capteurs virtuels, de se percevoir entre eux, mais aussi d'agir, de communiquer et de se coordonner entre eux. L'homme

participe à ces univers par le biais d'avatars. Ce sont des modèles numériques particuliers, dans le sens où leurs capacités de décision sont déléguées à l'opérateur humain qu'ils représentent.
La deuxième partie (au chapitre 3) présente les principaux modèles élaborés par l'auteur et son équipe, dans le respect du principe d'autonomie abordé au chapitre précédent. L'approche multi-agents appliquée au domaine de la réalité virtuelle, permet à l'auteur d'introduire la notion, très importante dans son approche, de simulation multi-agents participative, qui donne un rôle actif à l'opérateur humain dans la simulation. Les phénomènes d'auto-organisation collective, reproductibles dans les simulations multi-agents, conduisent l'auteur à défendre l'idée d'une expérimentation *in virtuo*, qu'il illustre dans une application à la biologie (la coagulation sanguine). Enfin, les cartes cognitives floues permettent la spécification du comportement des entités autonomes, mais aussi le contrôle de leurs mouvements et surtout donnent la possibilité aux entités elles-mêmes de simuler (dans la simulation) leurs mouvements afin d'anticiper sur leurs propres comportements potentiels. Cette utilisation des cartes cognitives floues modélise le comportement perceptif d'entités autonomes. Les modèles décrits dans ce chapitre seront implantés grâce aux outils proposés et décrits dans la troisième partie (au chapitre 4) du document.

Jacques Tisseau complète cette présentation synthétique des travaux de recherche qu'il a mené avec l'équipe qu'il dirige, par une notice individuelle composée d'un Curriculum Vitæ et d'une description de ses activités en matière d'encadrement, de publications, de suivi de contrats, d'enseignement et de responsabilités collectives.

L'originalité des travaux présentés s'appuie sur le concept d'autonomie et revient, dans les conditions qu'ils imposent, à la notion de réalité virtuelle pour qu'elle soit réaliste, c.-à-d. exploitable par l'homme. En effet la réalité virtuelle génère des univers virtuels qui doivent impérativement être composés d'entités (modèles numériques) *autonomes*, tout en incluant l'homme par le biais d'entités (modèles numériques) spécialisées appelées *avatars*. Dans ce contexte, un avatar n'est rien d'autre qu'une entité de l'univers virtuel, qui abandonne son propre contrôle (perte d'autonomie) à un opérateur humain. Ce faisant, cette approche permet à l'homme de rentrer dans l'univers virtuel (simulation) au même niveau conceptuel (modèles numériques) que les entités artificielles qui peuplent cet univers.
En s'appuyant sur le concept d'auto-organisation dans les systèmes multi-agents réactifs, ces travaux nous montrent également comment la réalité virtuelle peut devenir pour des domaines comme la biologie, un nouveau champ d'expérimentation (in virtuo) pertinent, à moindre coût, sans danger et très expressif (manipulation de modèles en cours de simulation).

Enfin, et pour illustrer la richesse et la puissance de l'approche, l'auteur montre comment la description de comportements émotionnels à l'aide de cartes cognitives

floues, conduit à ce qu'il appelle une ébauche de perception de soi (entité de l'univers virtuel), nécessaire à une véritable autonomie.
Tout ce travail de conceptualisation est concrétisé, donc viabilisé, par un très important travail de réalisation d'outils adaptés. Il s'agit principalement d'un langage et d'un simulateur appelés *oRis*, à la base d'une plate-forme de réalité virtuelle appelée *ARéVi*. La robustesse et la puissance de ces outils ont été largement validées par les différentes expérimentations qui ont eu lieu dans des domaines variés comme la biologie, les réseaux informatiques, l'éthologie, le traitement d'images, la simulation médicale ou encore manufacturière. On trouve dans le simulateur *oRis* l'idée d'un «prototypage interactif» par un procédé «d'immersion langagière» en référence à l'immersion multi-sensorielle de la «réalité virtuelle». Grâce à ce procédé fondé sur une approche langagière (au sens informatique du terme), qui repose sur le changement dynamique (en cours de fonctionnement) du comportement des composants impliqués dans les modèles numériques manipulés, il est donc possible de compléter les modèles existants par l'introduction de nouveaux modèles, ou de modifier les modèles en cours, afin de mieux comprendre leurs impacts sur le système étudié.

En dirigeant depuis son origine l'équipe qui a mené ces travaux, Jacques Tisseau a ouvert une voie prometteuse dans le domaine de la réalité virtuelle en y introduisant l'idée (principe) d'autonomie des entités qui peuplent les univers virtuels engendrés. Grâce aux outils présentés, l'auteur montre expérimentalement comment ce principe d'autonomie agit en terme d'auto-organisation collective sur les entités de ces univers, mais aussi au niveau de leurs perceptions individuelles dans ces univers.
Pour cela il a eu le mérite de rapprocher des résultats de recherches dans le domaine des systèmes multi-agents, du génie logiciel et de la réalité virtuelle, au sein d'une problématique qui dépasse chacun de ces domaines et qu'il nomme l'expérimentation *in virtuo*. En effet cette approche rend beaucoup plus accessible que par le passé, l'étude des systèmes complexes, caractérisés par une grande diversité des composants en jeu, une grande diversité des structures rencontrées et une grande diversité des interactions entre les nombreuses entités de ces systèmes, et pour lesquels, malheureusement nous ne disposons pas encore d'une bonne formalisation.

La juxtaposition des notions d'entité autonome et d'avatar au sein d'univers virtuels, permet à Jacques Tisseau de proposer la perspective du «principe de substitution», comme complément au «principe d'autonomie». Dans la mesure où les avatars et les entités autonomes ont la même structure, la substitution consiste simplement à déterminer dynamiquement qui, de l'utilisateur ou de l'entité elle-même, prend le contrôle du module de décision de cette entité.

Jacques Tisseau a construit un travail d'équipe dans le domaine de la recherche, qui est à la fois utile et courageux car, tout en assurant la responsabilité de la

filière informatique industrielle de l'ENIB, il a été le créateur du laboratoire d'informatique industrielle (LI2) de l'ENIB et son responsable depuis son origine (1990), mais aussi et surtout car il a su lui donner au cours du temps l'essor qu'il connaît actuellement (onze titulaires et six doctorants) avec comme principal actif, l'ouverture d'un CEntre de Réalité Virtuelle (CERV) à Brest en 2002/2003. Son travail de recherche au sein du LI2 a d'ailleurs été apprécié dans la communauté scientifique nationale, puisque le LI2 a été intégré à l'équipe d'accueil EA 2215 (UBO/ENIB) du Ministère de l'Éducation Nationale de la Recherche et de la Technologie, et qu'il est membre actif du groupe de travail Réalité Virtuelle (groupe transversal aux GDR I3, ALP et ISIS), du groupe EEA/SEE Téléproductique Bretagne et du groupe Architectures de Systèmes d'Agents de l'AFIA.

Jacques Tisseau a su faire partager les avancées des recherches du LI2 avec la communauté scientifique en étant responsable de l'organisation de quatre journées scientifiques nationales et d'un congrès international (avec édition des actes). L'équipe du LI2 a produit plus de cinquante-huit publications, dont une bonne partie dans des conférences nationales et internationales avec actes et comité de lecture et trois dans des revues nationales et internationales. Jacques Tisseau s'est directement impliqué dans le rayonnement du LI2, puisqu'il est expert auprès de l'INRIA de Rennes dans le cadre des appels d'offre sur la réalité virtuelle, ainsi qu'auprès de la Communauté Urbaine de Brest toujours pour ses compétences en réalité virtuelle.
Il faut souligner la qualité des travaux du laboratoire, qui permettent par leur aspect technologique fort (outil et méthode) de faire avancer des travaux plus fondamentaux dans des disciplines diverses et variées comme la biologie ou encore l'éthologie pour ne citer que celles-ci. Cela a permis au LI2 de réaliser neuf contrats industriels avec des partenaires institutionnels et industriels, conduisant à des prototypes et des rapports techniques associés.
Jacques Tisseau a également travaillé pour l'encadrement de jeunes chercheurs. Il a suivi quatre étudiants en stage de DESS, huit en stage de DEA et il a encadré de façon significative trois thèses de doctorat soutenues et trois autres en cours. Il a également participé à neuf jurys de thèse.
De façon évidente les documents fournis par le candidat attestent de son engagement décisif et constant dans la vie du laboratoire LI2 où il exerce son activité actuelle d'enseignant-chercheur. Les différents résultats obtenus par cette équipe qu'il anime depuis plus de dix ans, sont autant d'éléments indiscutables sur les qualités de Jacques Tisseau à diriger une équipe de recherche.

Pour toutes ces raisons je suis très favorable pour délivrer à Jacques Tisseau l'Habilitation à Diriger des Recherches, qui représente une reconnaissance méritée du travail qu'il a réalisé pour le LI2 et la communauté scientifique depuis plusieurs années.

François Bourdon
Professeur à l'Université de Caen.

Pour compléter le contexte des simulations dans lequel nous présentons la suite de nos travaux sur les heuristiques de résolution, il nous faut préciser les propriétés de l'outil *oRis*. Cet outil n'est pas un simple simulateur d'objets actifs ou d'agents autonomes comme il en existe beaucoup. Il garantit une très grande stabilité et une très bonne équité, tant dans la désignation que dans la gestion temporelle des processus parallèles sur une machine séquentielle classique.
Ces propriétés absolument indispensables sont malheureusement souvent oubliées, car elle nécessitent une très fine compréhension de l'interdépendance des mécanismes en jeu dans l'empilement des couches informatiques, particulièrement sensible dans le calcul parallèle distribué.
Comme le souligne Roger Cozien dans la conclusion du chapitre 4 de sa thèse intitulée «*Premiers Éléments de la Théorie du Calcul Singulier. Thermodynamique, dissipation logique et irréversibilité dans les systèmes informatiques logiquement distribués.*», parue en juin 2002:
«*cette très grande neutralité du moteur de simulation d'oRis est un acquis fondamental. Cela nous assure que les phénomènes singuliers que nous allons observer ne sont pas la conséquence, même lointaine, d'artefacts introduits, ou induits par oRis. De même, les résultats théoriques que nous pourrons extrapoler des expériences devraient pouvoir s'appliquer quelle que soit la plate-forme logicielle. Ainsi, nous pourrons dire que les phénomènes émergents observés sont indépendants des scripts oRis. Ils seraient alors imputables, soit à l'algorithme de référence dudit script - ce qui serait pour le moins ennuyeux, car il n'y aurait pas une totale indépendance entre l'algorithme de référence et la grammaire informatique utilisée pour l'implémenter -, soit au concept de répartition des calculs via un environnement d'interaction commun, ce qui serait l'explication la plus probable. Il y aurait alors une totale indépendance vis-à-vis du code, quelle que soit sa forme.*»

IV-C2) Propriétés d'*oRis*

L'essentiel des propriétés qui nous intéressent ici, c.-à-d. offertes à un outil tel *oRis* pour conduire des simulations sur l'émergence et la résolution heuristique de phénomènes répartis, décentralisés et autonomes, est repris avec des expérimentations dans le travail de thèse précité de Roger Cozien.
Plutôt que de paraphraser l'auteur, avec son accord, et sans reproduire ici les trente-deux pages de son manuscrit (au chapitre 4 «Environnement de simulation» p.93-125) consacrées à cette question, nous allons citer les passages les plus importants pour donner une idée précise au lecteur des attendus pour notre travail de simulation.

Il s'agit donc de montrer qu'*oRis* présente des caractéristiques d'équité logique et de désignation, mais également des caractéristiques d'équité tremporelle.

«…La première équité, si elle est vérifiée, assure que tous les OA[1], pour une unité de temps simulateur, seront désignés un même nombre de fois par l'ordonnanceur du simulateur. Vérifier cette équité logique, c'est vérifier que l'ordonnanceur n'introduise pas de priorité entre les objets, autre que celle que le programmeur pourrait avoir introduit volontairement.

équité logique

L'équité temporelle, quant à elle, doit se vérifier sur plusieurs niveaux. En effet, la simulation de processus parallèles sur des machines séquentielles impose de vérifier que les mécanismes de parallélisation logique n'introduisent pas de priorité entre les processus, et que, d'autre part, ils n'introduisent pas d'autres effets de bord ou «artefacts temporels», dont le principal est le phénomène de «distorsion temporelle». Ceci aussi bien lors de la comparaison entre la temporalité de l'observateur et celle de la machine, mais également dans la temporalité logique, lorsque le nombre de processus est variable. Le problème du temps, dépassant le cadre de l'équité, il est préférable de parler de «cohérence temporelle»…»

équité temporelle

distorsion temporelle

«…oRis propose un langage de programmation interprété développé à partir du GNU C++. Ici les objets actifs du langage sont très faiblement couplés avec les données qu'ils manipulent, à tel point qu'un objet peut solliciter l'exécution d'un traitement implanté dans un autre objet, avec des données issues du premier. L'autonomie introduite par les objets actifs autorise la réalisation d'applications décentralisées et pointe la nécessité de coordination des actions via des mécanismes de communication, voire de négociations entre ces objets actifs.
L'exécution de ces objets actifs est basée sur la notion de thread (fil d'exécution). Ces threads partagent l'activité du ou des processeurs physiques de la machine. Ceci signifie qu'il faut pouvoir faire la différence entre l'ordonnancement de la machine et celle choisie dans le simulateur via la structure du code et la logique décrite par l'expérimentateur.

Il apparaît que les phénomènes émergents doivent exister au-delà du code et des algorithmes utilisés. En effet les modèles de la programmation distribuée ont la particularité d'induire un style de programmation basé sur la déconcentration de la fonction logique du programme de référence.
Pour obtenir cela, chaque objet actif possède son comportement propre et autonome, qu'il embarque dans son code et qu'il exécute perpétuellement dès qu'il a été instancié dans la simulation.
Tous les objets actifs exécutent leur code dans le même flot temporel logique (top_i vu précédemment).
oRis propose trois modes de gestion des threads (parallélisme) appelés coopératif, préemptif et massivement parallèle.
Le premier exécute l'intégralité du comportement (méthode main) de chaque objet actif, quelque soit la durée de ce dernier. Le second préempte (interrompt) l'exécution des comportements des objets actifs passé un délais fixé par

1 Objets actifs.

le programmeur. Alors que le troisième se base, non pas sur une mesure du temps processeur, mais sur un nombre de micro-instructions du moteur de simulation.

Le premier mode coopératif peut introduire des blocages ponctuels de la simulation lorsque des comportements d'objets actifs sont très disparates en durée. D'autre part le temps exact alloué par le simulateur à chaque instance est potentiellement différent puisqu'elle peut rendre la main quand elle veut. Nous avons essentiellement utilisé ce mode dans nos simulations, puisque tous nos objets actifs ont des comportements analogues en durée. Les simulations sont donc très fluides et très rapides puisqu'elle ne requièrent pas de mécanismes supplémentaires de préemption, comme c'est le cas pour les deux autres modes et tout particulièrement le troisième.

Dans le mode préemptif une instruction du simulateur permet d'indiquer l'intervalle de temps alloué à chaque thread (flot d'exécution) de chaque instance. À la fin de ce temps il y aura automatiquement préemption vers l'instance suivante. Ce mode assure donc l'équité temporelle entre les objets actifs, qui de cycle en cycle, seront actifs pendant la même durée de temps.

L'organisation induite par les objets actifs de ce que l'on peut appeler un code multi-activités, aboutit à une exécution quasi-parallèle des sous-fonctions de l'application, résultant d'un éclatement de la fonction principale dans plusieurs objets actifs…»

«*…Toutes ces nouvelles possibilités ne sont pas gratuites. Outre la nécessité d'un ordonnanceur, il faut assurer la cohérence d'évolution entre les objets actifs. Comme ils sont autonomes nous parlons d'ailleurs de co-évolution. L'autonomie intrinsèque des objets actifs, présente dès la phase de conception, empêche la conception d'un objet supérieur contrôlant les autres instances. La coordination de la co-évolution n'a d'autres médias que les communications entre objets actifs. Ces communications peuvent revêtir plusieurs formes : l'appel, ou l'invocation, d'une méthode d'un objet actif distant, l'envoi de messages dans une boîte aux lettres, l'envoi de messages par diffusion radio, les liens réflexes, la modification de l'environnement.*
La communication entre objets actifs, combinée avec la possibilité pour chaque objet de refuser la demande qui lui est faite, doit normalement assurer la cohérence de la co-évolution, et donc, la cohérence de l'exécution de l'application. Pourtant, l'utilisation d'un modèle de programmation plus haut dans la hiérarchie apporte plus d'expressivité, mais cette facilité n'est pas gratuite, même si ce coût échappe à la grande majorité des programmeurs. Ce coût est contenu dans ce qui, «d'explicite» dans un modèle donné, devient «implicite» dans le modèle immédiatement supérieur. Cette transformation de l'explicite vers l'implicite est en partie responsable des phénomènes émergents dans les SILD - «systèmes informatiques logiquement distribués…»

L'utilisation des objets actifs

«…L'autonomie des instances dans oRis est triple. Il y a l'autonomie d'exécution, l'autonomie fonctionnelle et l'autonomie de code.

L'autonomie d'exécution passe par un flot d'exécution (thread) attaché à chaque instance, permettant un comportement proactif, par opposition à l'invocation explicite des méthodes, présente également dans oRis.

L'autonomie fonctionnelle des objets actifs se concrétise par le fait qu'un objet actif peut décider de refuser l'accès à l'une de ses méthodes par un autre objet actif. Même si l'on ne peut pas empêcher un appel explicite de méthode, il est possible dans oRis d'identifier l'instance à l'origine de l'appel et d'interdire l'exécution de la méthode.
Enfin l'autonomie de code repose sur un langage interprété dont le code est dynamique. Cela permet, par un opérateur humain, lors de l'exécution de simulations, de créer de nouvelles classes et de nouvelles instances, de surdéfinir une classe, une méthode de classe, une méthode d'instance. On pourra également ajouter des méthodes à une instance…»

«…Il est donc possible de différencier une instance de sa classe d'origine. Cet objet actif devient alors l'instance d'une classe qui n'a pas été écrite dans le code d'origine. Ce premier aspect de l'autonomie de code n'est que le précurseur d'un autre plus profond. Il existe dans oRis un ensemble de commandes permettant d'une part l'introspection du code d'une instance par elle-même ou une autre, et d'autre part, la manipulation et donc la modification du code de l'instance…»

Moteur de simulation et équité

«…Une fois les instances d'objets actifs créées, l'ordonnanceur d'oRis à deux modes de désignation de ces instances. Le premier est séquentiel. Il désigne toutes ces instances dans un ordre immuable d'un cycle à l'autre, à l'image du parcours d'un tableau en 'C'. Le deuxième est aléatoire. Il repose sur une désignation aléatoire, sans remise (chaque instance ne peut être choisie qu'une seule fois par cycle), d'un cycle à l'autre.
Nous n'utilisons que le mode aléatoire dans nos simulations afin d'éviter toute priorité implicite (créant des effets de bord potentiels) entre l'ordre d'exécution des instances. Ce mode assure un brassage des instances et instaure un principe d'équité des instances face au moteur de simulation.
Pour obtenir cela Fabrice Harrouet, le concepteur d'oRis, a proposé un algorithme basé sur deux tableaux le premier appelé émetteur et le second appelé récepteur, de dimension 'N', le nombre d'instances dans la simulation. Le simulateur place les instances dans le premier tableau, suivant l'ordre d'instanciation de ces objets actifs. Ensuite il tire un nombre aléatoire $\alpha \epsilon [0,1]$ *et le compare à ½. Si* $0 \leq \alpha \leq ½$ *alors le simulateur choisit l'extrémité 'A' du*

deuxième tableau récepteur, sinon il choisit l'extrémité 'B' de ce tableau pour placer l'instance courante.
Puis l'instance choisie est activée, son code s'exécute, et elle est placée dans un deuxième tableau vide au rang i=0. On réitère. À ce moment, le simulateur choisit une nouvelle instance placée à une des extrémités du premier tableau qui ne contient plus que N-1 éléments. La deuxième instance est finalement placée dans le deuxième tableau au rang i=1. Cet algorithme est exécuté tant qu'il reste au moins une instance dans le premier tableau. Le placement de la dernière instance du premier tableau vers le deuxième clôture un cycle de simulation. Le cycle suivant consiste à inverser les rôles des tableaux émetteur et récepteur. Ce mode de désignation est équivalent, après quelques cycles, à une désignation par tirage aléatoire sans remise…»

Roger Cozien s'appuie sur le théorème des dérangements pour justifier cette propriété d'équité entre le tirage des objets actifs d'*oRis*. Dans son étude, il a vérifié expérimentalement ce principe, en notant que cet aléatoire construit, intervient aussi bien dans le moteur de la simulation que dans le comportement des objets actifs.
Pour cela il a conduit plusieurs expériences dont l'une sur la répartition homogène de points sur des cercles, une autre sur la simulation du front d'onde en physique, qui met en œuvre une répartition aléatoire, et enfin la simulation de la croissance de l'entropie, toujours dans le contexte d'expériences physiques réelles.

«*…Tous les résultats de ces expérimentations convergent vers une même conclusion, en l'occurrence que l'ordonnanceur et le générateur pseudo-aléatoire d'oRis, et donc le moteur de simulation, sont équitables et que l'on peut sereinement envisager, tant la simulation de phénomènes réels, que l'exécution de n'importe quel code distribué, malgré la séquentialité du support électronique…*»

NEUTRALITÉ TEMPORELLE DE L'ENVIRONNEMENT DE SIMULATION

Roger Cozien aborde dans cette partie la question cruciale de l'équité de la distribution du temps aux différentes instances de la simulation.

Pour cela il définit un protocole expérimental afin de valider cette propriété dans les deux modes de désignation (séquentiel et aléatoire) des objets actifs, ainsi que dans les trois modes multitâches (coopératif, préemptif et massivement parallèle). L'objectif est de vérifier que la distribution du temps dans les durées visées, est linéairement proportionnelle à l'augmentation du nombre d'instances.

«*…À l'issue de ces expérimentations (très grand nombre d'expériences et de mesures), l'auteur affirme la très grande neutralité du moteur de simulation. Ceci est dû grâce à l'environnement logiciel offert par oRis qui est éminemment stable, mais surtout, équitable, tant dans la désignation que dans la gestion temporelle des processus parallèles.*

Cette très grande neutralité du moteur de simulation est un acquis fondamental. Cela nous assure que les phénomènes singuliers, que nous allons observer, ne sont pas la conséquence, même lointaine, d'artefacts introduits, ou induits par oRis…»

Simulations : le code

IV-D

Comme introduit précédemment, le principe de base des heuristiques proposées est de se placer dans un processus auto-organisé, décentralisé, capable d'adaptation à un environnement changeant, sans être prévisible au-delà d'un certain horizon.
En nous appuyant sur les résultats obtenus dans l'étude théorique de C_nS_k (au chapitre III), à savoir les notions de *k-partition* «utile» et de multi-points MM_{k-p} «utile», nous adoptons, pour le code des expérimentations, les principes généraux suivants :

- le système se compose d'un ensemble 'C' de clients cl_i, de serveurs $serv_j$ et de requêtes req_l ;
- nous faisons l'hypothèse que les clients sont immobiles ; dans ce contexte leur position, leur demande, ainsi que leur nombre, possiblement variable, font intégralement partie de la description du problème C_nS_k à résoudre ;
- utilisant le simulateur *oRis* (cf. la partie IV-C), le temps du système est rythmé par celui des pulsations (top d'horloge) du simulateur, permettant aux acteurs, dans la simulation, de se synchroniser ;
- les demandes des clients correspondent à une quantité de ressources nécessaire à chaque top d'horloge. Ces demandes ne pourront évoluer que sur des changement de top d'horloge ;
- les serveurs sont les entités capables de mobiliser les ressources de l'environnement (par exemple de la puissance de calcul) pour mettre en œuvre des réponses aux besoins des clients. Ils sont mobiles et se déplacent de façon continue dans l'environnement à l'image d'un réseaux fortement maillé par des nœuds de calcul.

L'un des objectifs est de mettre en relation les clients et les serveurs dans une résolution individuelle et collective au plus juste, sans aucun contrôle centralisé. Il s'agit de satisfaire tout le monde en minimisant les ressources mobilisées dans l'environnement.
Dans la mesure où les nœuds (incluant l'infrastructure de communication) font partie de l'environnement, on pourrait introduire leur dimension fonctionnelle dans la résolution du problème. Nous ne l'avons pas fait dans nos simulations pour nous concentrer sur les dynamiques en jeu.
En revanche nous l'avons fait avec Hugo Pommier et Benoît Romito dans leurs thèses respectives. En effet le stockage des documents sous la forme de nuées d'oiseaux, toujours en mouvement dans le réseau, prend en compte des propriétés de clustering (probabilité pour des équipements de tomber simultanément en panne…) des machines (nœuds) et des liens entre elles.
En partant de ces remarques, nous allons décrire dans cette partie IV-D comment sont codées les heuristiques qui seront à la base des résultats expérimentaux rapportés dans la partie IV-E suivante.
Nous allons détailler la structure du code de la simulation, avec son organisation et la façon dont elle pourra être paramétrée pour conduire les expérimentations. Nous aborderons également la chaîne d'outils réalisée dans ce contexte pour visualiser certaines propriétés des expérimentations.
Enfin nous aborderons des points techniques particulièrement éclairant pour nos analyses ultérieures.

Avertissement : tout au long de cette partie et de la suivante nous pourrons utiliser le terme «la simulation» pour désigner le code réalisé sur le simulateur *oRis* dans le cadre de cette étude. Dans la mesure où ce code générique est paramétrable, comme nous le détaillerons par la suite, nous parlerons «d'expérimentations» pour désigner les simulations faites avec «la simulation». Chacune d'elles correspond donc aux résultats de l'exécution de «la simulation» avec une configuration donnée de ses paramètres.

IV-D1) L'ORGANISATION GÉNÉRALE DU CODE DES SIMULATIONS

Les simulations fonctionnent sur le principe général d'*oRis* avec une boucle infinie, basée sur la notion de top (logique) d'horloge. À chaque top, l'ensemble des objets actifs créés, exécutent leur propre fonction interne *execute*, associée à leur comportement dans l'environnement choisi (à deux ou trois dimensions). Grâce à ces principes les objets actifs peuvent, de façon parallèle et autonome, évoluer, communiquer, apparaître, disparaître,…
Nous distinguons donc «la simulation», qui correspond au code écrit sur *oRis* pour réaliser nos expérimentations, du code nécessaire à chacune de ces expérimentations.
Une expérimentation repose sur une configuration (choix de constantes) donnée de «la simulation». Le code d'une expérimentation crée les objets actifs nécessaires, ainsi que les fenêtres visualisant, sous forme de graphiques et en temps réel, l'évolution de certains paramètres choisis.
Le moteur *oRis* lance les expérimentations et permet de suivre en temps réel leur évolution (cf. la figure *Fig-IV-D1-33*).

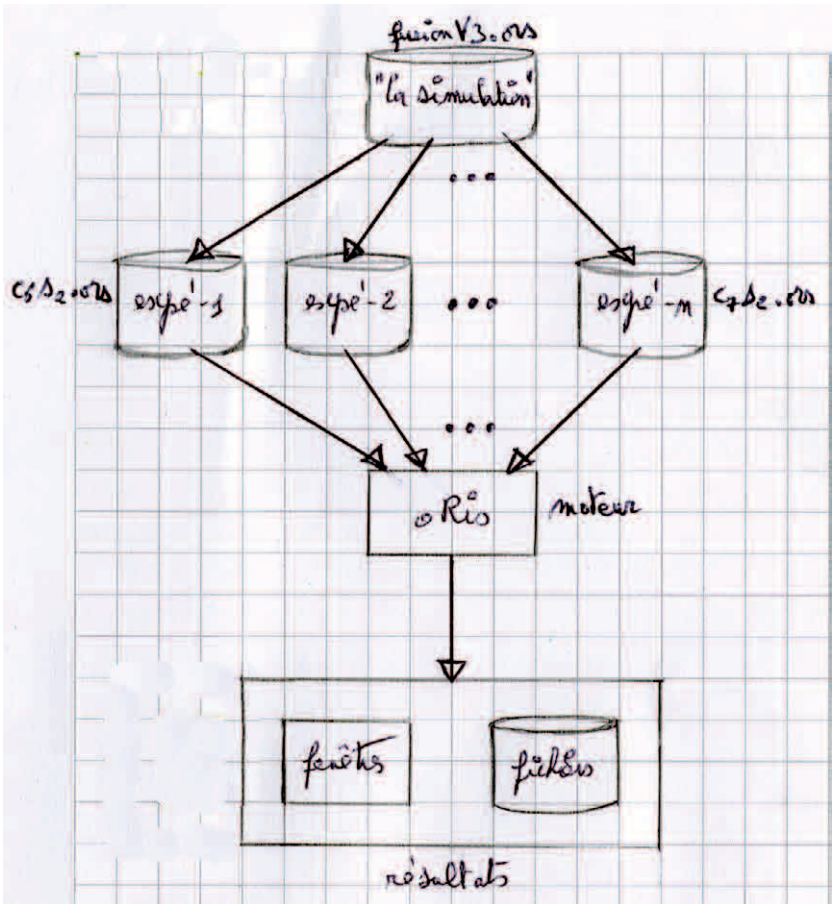


Fig-IV-D1-33

Le code de «la simulation», contenu dans le fichier «fusionV3.ors», est organisé de la façon suivante :

```
// inclusion des packages oRis
include "maths.pkg"
include "object2d.pkg"
include "grapher.pkg"
include "convert.pkg"
include "files.pkg"

// inclusion des constantes de la simulation
include "fusionV3.cstes"

// inclusion des définitions des classes de la simulation
include "fusionV3.def_classes"

// inclusion du code des classes de la simulation
include "Manager.ors"
include "DomaineManager.ors"
include "Serveur.ors"
include "Requete.ors"
include "Client.ors"
include "grille.ors"

// programme principal
execute
{
   setPickingMode(randomPicking);
}
include "initGRILLE.ors"
```

Nous trouvons d'abord l'inclusion de différents packages proposés par *oRis*, qui nous serons utiles ici. Ceci montre la modularité du simulateur avec un noyau et des packages additionnels et donc optionnels.
Nous avons construit le code de la simulation en séparant les constantes (fichier «fusionV3.cstes»), les définitions des classes (fichier «fusionV3.def_classes») et le code des classes écrites pour la simulation (un fichier par classe).
On peut voir, dans le programme principal de «fusionV3.ors» l'appel à la fonction *execute* dont le rôle est juste de positionner le mode de choix de l'ordre d'exécution des objets actifs. Ici nous utilisons le mode aléatoire qui nous garantit un ordre aléatoire dans l'exécution de chaque objet actif à l'intérieur de chaque top logique du moteur *oRis*.

Les constantes de la simulation

Les constantes (plus de soixante) sont en réalité des grandeurs partagées qui configurent la simulation en cours.

Elles donnent à la simulation des propriétés comportementales objectives qui permettront d'isoler des phénomènes.
L'outil nous permettraît donc d'aller plus loin dans les expérimentations en jouant sur l'évolution dynamique de ces «constantes» (modifier leur valeur en cours de simulation). Nous avons pu le faire localement.

Cela voudrait dire que la simulation serait polymorphe et surtout capable de changer de forme (d'algorithme) en fonction des objectifs recherchés.

Bien que, techniquement, rien n'interdise cette approche, dans ce cas nous serions confronté à une difficulté immense pour comprendre et analyser les résultats. Sachant qu'il est tout aussi important de comprendre comment on arrive à ces résultats.

Nous n'allons pas énumérer toutes les constantes définies dans «fusionV3.cstes», mais en dégager les contours, afin d'avoir une idée du potentiel de configuration de nos expérimentations.

Certaines d'entre elles permettent de piloter les algorithmes/fonctionnement de certains objets actifs :

- *rechercheAutoPO* : si elle est *true*, lancera, chez les serveurs, une recherche automatique de la période d'observation (PO) en utilisant les pas d'exploration, sinon la PO des serveurs restera fixe/constante. Nous reviendrons en détails sur cette notion de période d'observation (PO) pour comprendre comment elle se cale sur les tops d'horloge du moteur *oRis* ; rechercheAutoPO
- combinée avec *avecCooperation*, mise à *true*, cette recherche automatique de PO sera faite en coopération avec les autres serveurs. Plusieurs constantes permettent de piloter l'évolution des PO de chaque serveur dans les phases d'exploration et de recherche de la PO optimale ; avecCooperation
- on peut également définir la stratégie de déplacement des serveurs en fonction de la distance au but visé ;
- on peut définir à quel moment (chaque top d'horloge ou en fin de PO) les serveurs vont produire des ressources disponibles. Là encore nous pourrions mettre ces valeurs propres à chaque serveur, autorisant des comportements différents. Cela rendrait trop complexe l'analyse des résultats. D'un autre côté, le fait d'avoir des comportements communs pour des instances d'une même classe, n'est pas non plus irréaliste ;
- certaines constantes définissent la précision utilisée lors des calculs du positionnement des serveurs sur une grille discrétisée dans l'environnement de la simulation ;
- on peut régler la vitesse de déplacement des requêtes vers les serveurs. La valeur de la constante *INV_PAS_DEPL_REQ* correspond au nombre de tops d'horloge pour parcourir une unité de distance dans l'environnement ; INV_PAS_DEPL_REQ
- on peut aussi choisir le mode de visite des serveurs par les requêtes. Soit une requête ne peut plus visiter un serveur déjà visité (ce qui impose une mémorisation par la requête des serveurs déjà visités), soit il le peut, mais pas deux fois de suite… ;
- on peut faire le choix de calculer une distance du serveur courant au client visé uniquement sur la base du calcul euclidien, ou en pondérant cette valeur par la ressource demandée par le client ;
- …

Comme nous le verrons par la suite, le *Manager* est un objet actif qui centralise le résultat de certaines actions des objets actifs. Il n'agit absolument pas sur les comportements des objets actifs dans la simulation, mais recueille des informations pour les afficher en temps réel sous forme de graphiques et/ou de données stockées dans des fichiers.
On retrouve exactement la même idée qu'avec la base de données stockant des informations sur le système de fichier dans *Linux*, ou encore le pseudo système de fichier «/prod», toujours sur *Linux*, qui stocke en temps réel des informations sur l'état du système.
Toutes ces informations sont consultables, mais non nécessaires pour le fonctionnement du système lui-même.

Certaines constantes permettent ainsi de jouer sur des éléments visuels de la simulation, comme par exemple, lors des déplacements des serveurs, où l'on affiche un cercle autour de chaque client, matérialisant la distance au premier serveur vu.

Les classes de la simulation

Le simulateur *oRis* propose, pour coder les simulations, un langage orienté objet propre, basé sur le langage *C++,* dans lequel la notion de pointeur n'est plus explicite. Les particularités spécifiques de ce langage sont multiples, puisqu'il intègre son propre ordonnanceur dont nous avons déjà évoqué les trois modes possibles (*coopératif, préemptif et massivement parallèle)*, qu'il est interprété et qu'il permet une gestion introspective du code pendant son exécution. Cette introspection va très loin. En effet on peut examiner et modifier à l'exécution toutes les variables, mais aussi toutes les méthodes des classes des objets actifs. Cette dernière propriété est fondamentale pour faire du prototypage dynamique ou incrémental. Ce n'est pas la simulation, elle-même, qui modifie son propre code en cours d'exécution, mais l'expérimentateur. Cela évite de rejouer la simulation depuis le début, tout en modifiant son comportement.

«La simulation» est donc écrite avec des classes dont nous avons séparé, dans des fichiers distincts, les définitions des méthodes et leur code. Nous avons dix-huit classes avec environ deux-cent-soixante-six variables de classe, dont de nombreuses sont des tableaux dynamiques, et cent-quatorze méthodes de classe, pour un total de quatre-mille-vingt-sept lignes de code (version v3).

Certaines classes jouent un rôle plus important que d'autres à la fois en terme de volume de code, mais aussi dans la nature active qu'elles donnent aux objets correspondants. En fait dans *oRis* nous pouvons manipuler deux types d'objets. D'un côté nous avons les objets classiques, d'un autre côté les objets actifs. Un objet actif possède une méthode *main*, que les autres objets classiques n'ont pas. Cette méthode permet de décrire le code qui sera activé à chaque top d'horloge par le moteur *oRis*. Dis autrement, ces objets actifs ont un comportement propre, proactif.

Cela ne les empêche pas, comme les autres, d'être invoqués par d'autres objets (actifs ou non) pendant le déroulement des simulations. Tout objet (actif ou classique) peut posséder dans ses données internes des références sur d'autres objets.

Nous allons présenter les cinq plus importantes classes d'objets actifs, sur les sept existantes dans la simulation, à savoir les classes *Manager*, *DomaineManager*, *Serveur*, *Client* et *Requete*:

- Classe *Manager*: une seule instance de cette classe est créée au début de chaque expérimentation. Elle configure la simulation en fonction des constantes du fichier «fusionV3.cste». Elle ouvre les canaux vers les fichiers de résultats. En fin de chaque période d'observation, définie par un nombre de tops d'horloge, elle centralise les informations en provenance des objets actifs pour pouvoir alimenter l'affichage temps réel de ces valeurs dans les fenêtres graphiques. Une simulation peut être divisée en plusieurs sous-domaines. Chaque sous-domaine, comprenant des clients et des serveurs, est géré par une instance de la classe *DomaineManager*, qui lui est propre. Toutes ces instances de la classe *DomaineManager* sont gérées par la seule instance de la classe *Manager*. Le *Manager* ne gère donc aucun client ni aucun serveur en direct. Par cette technique la simulation peut dynamiquement faire apparaître ou disparaître des sous-domaines. Ce sera le cas dans l'algorithme de mitose (division cellulaire) où un serveur, appartenant à un sous-domaine, peut décider de se dupliquer (méthode *fork_srv*) en créant un nouveau sous-domaine afin de partager le sous-domaine courant en deux sous-domaines (mitose). Dans ce cas le *Manager* gère les demandes de duplication de serveurs faites par les serveurs eux-mêmes, sous la forme d'une liste de candidats à la duplication. L'idée étant d'optimiser la création de serveurs en évitant des créations inutiles. Pour cela des seuils sont utilisés. Pour chaque candidat serveur à la duplication, on dispose de sa densité d'échec (nombre d'échecs dans une période d'observation) et du volume de ressources demandées par des clients qu'il n'a pu fournir, toujours sur la dernière période d'observation. On peut ainsi trier la liste des candidats serveurs à la duplication par densité d'échecs décroissante… Un calcul du débit initial (quantité de ressources produites) sera effectué pour le serveur dupliqué en tenant compte du volume d'échecs par le géniteur. Enfin une vérification est faite afin que cette nouvelle capacité de production, ajoutée aux existantes, ne provoque pas un dépassement de la capacité maximale du sous-domaine concerné. L'algorithme peut également gérer des stratégies de positionnement des serveurs au moment de leur création. Le *Manager* (unique instance) peut lancer l'écriture des graphes d'états (méthode *sGSALL*) des différents *DomaineManager* de l'expérimentation. Nous reviendrons en détails sur la notion de graphe d'états d'une expérimentation. Il s'agit d'analyser la dynamique du système via l'expérimentation. La matrice des états code le passage des états entre eux. Un état, au temps

Manager

't' est défini par la répartition des clients sur les serveurs d'un sous-domaine donné. À t+dt l'état peut avoir changé. L'idée est donc de créer un arc dans la matrice entre ces deux états que l'on va considérer comme adjacents;

DomaineManager

- Classe *DomaineManager*: comme indiqué plus haut, un *DomaineManager* gère un ensemble de clients et de serveurs. Il a dans ses données d'instance, les listes des clients et des serveurs gérés (rangées par type). Il possède les méthodes de création de client (*creer_Client*) et de création de serveur (*creer_Serveur*), ainsi que celles de destruction de ces entités. Dans la simulation il existe donc deux façons de créer un serveur. Soit au lancement de la simulation en passant par le *Manager* qui crée le premier *DomaineManager*. Ce dernier va créer des serveurs. Soit dans l'algorithme de mitose où un serveur peut demander la création d'un nouveau serveur. Dans ce cas il s'adresse au *Manager* pour avoir l'autorisation, et lance une création via son *DomaineManager* ou un nouveau *DomaineManger* qu'il va faire créer par le *Manager*. Le moteur *oRis* offre un mécanisme pour découvrir l'objet le plus proche dans l'environnement (méthodes *view* et *viewFisrt*). Nous n'utilisons pas ces méthodes, car elles ne permettent pas de cloisonner l'espace de simulation, comme nous le permet les notions de *Manager* et de *DomaineManager*. Ce sont les requêtes qui, une fois créées par les clients, vont rechercher les serveurs les plus proches en utilisant la méthode *view_agt* du *DomaineManager* correspondant à leur client. Cela permet effectivement de récupérer le serveur le plus proche dans un sous-domaine donné, même si un autre serveur était encore plus proche, mais dans un autre sous-domaine. Nous verrons dans l'algorithme de mitose, comment cela permet de restructurer une zone stable sans déstabiliser les zones adjacentes. Une autre fonction du *DomaineManager* est de gérer son graphe des états. Si la constante *TRACE_ETAT_DM_GS* est à *true*, alors, de façon proactive, le *DomaineManager*, dans sa méthode *main* (à chaque top de la simulation), met à jour la matrice des états (méthode *calcul_matrice_etats*), en vérifiant si l'état en cours existe déjà ou non. Cette matrice (cf. la figure *Fig-IV-D1-34*) représente les arcs existants entre les états du système, c.-à-d. quand le système passe d'un état à un autre. Par exemple l'état d'indice 1 dans le tableau *Etats*, qui s'appelle A_1, sur la figure *Fig-IV-D1-34*, correspond à l'arc $Arcs_1$, qui traduit les passages possibles de A_1 vers A_5 et de A_1 vers A_7. Nous rappelons qu'un état correspond à la répartition des clients du sous-domaine sur les serveurs du même sous-domaine. Cette matrice va être utilisée par la librairie *GraphStream* (sur *Java*) pour afficher le graphe de ces différents passages entre états du système. On pourra ainsi analyser la dynamique du système. La matrice est symétrique et la diagonale est nulle. Si,

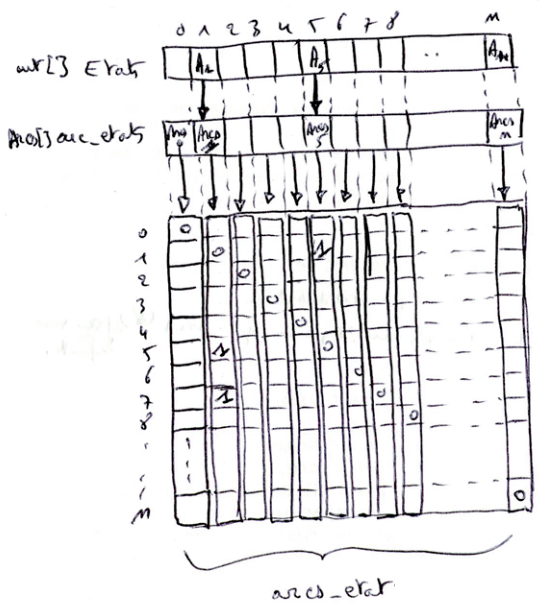


Fig-IV-D1-34

pendant l'exécution du système, plusieurs occurrences d'un arc apparaissent (le système est passé plusieurs fois d'un état donné à un autre), on ne gardera que la valeur de 1 dans la matrice, car *GraphStream* ne sait pas exploiter une telle valeur supérieure à 1. En revanche cette valeur sera conservée dans la variable *arc_etats* et utilisée à l'affichage comme label sur l'arc tracé par *GraphStream*. Pour des questions de performance et de cohérence, il faut coder chaque état (*Etats*) sous la forme d'un entier avec comme contrainte de ne pas coder des entiers différents pour des états identiques. Nous verrons en détails comment cela est réalisé dans le code, dans la partie *IV-D2*. Une simulation sur le moteur *oRis* tourne indéfiniement jusqu'à ce que l'opérateur l'arrête, de façon externe, ou, de manière interne, qu'un évènement particulier dans la simulation se produise, entraînant cet arrêt. Dans notre cas, seul l'opérateur peut l'arrêter, puisque l'objectif de la simulation est de rechercher des états stables qui proposent des solutions opportunes au problème posé (C_iS_j). Dans ce contexte la matrice des états ne sera sauvegardée sur disque uniquement de façon externe, c.-à-d. par l'opérateur dans la fenêtre d'exécution de code après avoir mis en pause la simulation. Il lui suffira de lancer la méthode *sGSALL* du *Manager* («Manager.1->sGSALL();») qui appelle la méthode *sGS* de chaque *DomaineManager* afin d'écrire sur disque (sortie *fc11*) les données de chaque graphe des états;

- Classe *Serveur*: c'est l'une des classes les plus importantes de la simulation. Les serveurs sont mobiles afin de se rapprocher des positions optimales dans l'environnement (satisfaction optimale des besoins des clients). Ils peuvent aussi se dupliquer, lorsqu'ils sont «stables», en créant éventuellement de nouveaux sous-domaines (algorithme de mitose). Le fonctionnement de la méthode *main* se décompose en deux parties: soit le top d'horloge courant correspond au dernier d'une période d'observation (PO), soit c'est un top ordinaire. Dans ce second cas le serveur calcule son déplacement (constante *RESTRUCT* à *true*) dans la direction choisie en fin de PO, avec la méthode *aller_vers*, si la descente de gradient n'est pas activée. Cette méthode *aller_vers* fait varier le rapprochement du serveur (dx/dy) vers sa cible de façon inversement proportionnelle à sa distance à cette cible. Si la descente de gradient est activée, alors le serveur utilisera la méthode *calculCible_allerY*. Cette méthode recherche la position vers laquelle le serveur courant va se déplacer en utilisant la descente de gradient. Quatre directions sont testées en commençant par celle de l'angle courant du serveur (*theta_cour*), puis theta_cour+90°, puis theta_cour+180°, enfin theta_cour+270°. Pour chaque direction, on calcule le coût total du serveur à ses clients (*CGoS*), en arrêtant la recherche pour la première direction qui propose un coût inférieur au coût correspondant à la position courante du serveur. Si aucune direction ne propose mieux, le déplacement s'arrête, le serveur a atteint un minimum local. Si une direction différente de la direction en cours propose un gain, cela signifie que le serveur changerait d'état

Serveur

en allant à cette position. Enfin si tous les gains des différentes directions sont négatifs, en particulier la direction courante, cela signifie que le serveur est dans le minimum local et ne peut que remonter vers la crête du sous-état courant. Une fois déplacé, le serveur va augmenter ses réserves de production pour le top suivant. Dans le cas du dernier top d'horloge de la période d'observation (PO), le comportement du serveur est tout autre. Il commence par vérifier qu'il travaille avec au moins un client, sinon il s'auto-détruit. Il va également mettre à jour son ajustement de production pour la PO suivante, pour mieux coller à la demande dans le futur immédiat. Il peut décider de se réorganiser à la suite d'un fonctionnement local qu'il jugerait non satisfaisant. Plusieurs méthodes peuvent être utilisées, par le serveur, pour détecter ce disfonctionnement, par exemple il peut se baser sur le nombre d'échecs de traitement des requêtes sur la dernière période d'observation, ou encore sur la valeur moyenne de variation de l'offre par rapport à la demande («ajustement»). Ensuite il calcule sa nouvelle direction à prendre (méthode *restructurer*) qu'il suivra à chaque top de la prochaine période d'observation (PO). Plusieurs heuristiques peuvent être choisies en fonction du moment de la simulation et de ce que l'on cherche à tester. La méthode classique est l'algorithme de *contraction/expansion* (C/E). Il tient compte d'un meilleur placement des serveurs pour optimiser le coût global de communication entre les clients et les serveurs. Dans cet algorithme, la *contraction* consiste, pour un serveur en échec (un volume de demandes plus important de la part des requêtes, que sa capacité à produire sur une PO) à se déplacer vers sa ou ses plus grandes demandes (clients). L'objectif est déliminer ,autant que possible, les demandes marginales pour quelles aillent vers d'autres serveurs, minimisant ainsi ses propres échecs à venir. À l'inverse, l'*expansion* offre la possibilité au serveur sous-employé (une production supérieure à la demande) de gagner des interactions indirectes. Ce sont des demandes issues de requêtes qui ont échoué avec un premier serveur cible. Pour cela le serveur se dirige vers des clients qui sont à l'origine de ses interactions indirectes. On peut aussi choisir un marcheur aléatoire parmi les clients vus, afin d'explorer davantage l'espace des états du système. On peut encore utiliser un marcheur aléatoire vers n'importe quelle position du sous-domaine courant (*DomaineManager*). Au départ, le serveur est dans une phase d'exploration du sous-domaine courant; pour cela il se déplace «rapidement» vers un client du domaine tiré au hasard, parmi ceux qu'il connaît. À chaque échange/interaction avec un autre serveur, le serveur courant met à jour ses connaissances du sous-domaine (enveloppe de recouvrement et liste des clients). Ensuite, le serveur regarde s'il est dans un état stable, qui correspond à une production constante sur une certaine durée et à d'autres critères internes. Dans ce cas plusieurs stratégies peuvent être déployées (méthode *agir_stable*), en particulier la recherche de la position optimale dans l'attracteur courant (état

courant). C'est dans cette situation que les serveurs n'arrivant pas à calculer la position optimale, vont tenter une division cellulaire (mitose) en appelant la méthode *fork_srv*, dont nous avons parlé plus haut. Sans entrer dans les détails, cette fonction est appelée par le serveur courant qui a atteint un équilibre durable et qui ne peut calculer (trop grande complexité) la grille optimale pour la sous-région «virtuelle» qu'il vient de faire apparaître. En effet, avant de se cloner, le serveur procède à la mise en place d'un sous-domaine inclus dans son domaine. Ceci permet de rendre étanche les réorganisations dues à sa duplication vis-à-vis des autres régions en équilibre. Dans le cas contraire (instabilité) le serveur va chercher à coopérer avec des serveurs dans son voisinnage qui sont stables. Là encore, plusieurs stratégies sont possibles (méthode *agir_instable*) pour récupérer la valeur de PO d'un ou de plusieurs (calcul de moyenne) serveurs stables. L'idée étant de propager des valeurs de PO des serveurs stables vers les serveurs instables;

- Classe *Client*: les clients ont des besoins en ressources qu'ils expriment en générant des requêtes mobiles et furtives, dont le rôle sera de trouver le ou les serveurs susceptibles de satisfaire leurs besoins. Si l'on prend l'exemple de la traduction de documents, les clients ont des documents à traduire, les requêtes emportent ces documents vers des serveurs qui mobiliseront les ressources nécessaires dans le réseau pour réaliser ces traductions. À chaque top d'horloge du moteur, si la constante *TRACE_CLIENT_CERCLE* est à *true* dans le sous-domaine où se trouve le client, alors une recherche (méthode *view_agt*) du serveur le plus proche permettra de tracer un cercle centré sur le client et passant par le serveur trouvé. Ensuite la client génère une requête avec un type de serveur visé (variable *montype*). Cette dernière possède la quantité de ressources demandée pour le futur serveur cible, ainsi que les coordonnées du client. Des interactions vont avoir lieu entre le client et la requête, à chaque fois que cette dernière atteindra un serveur; Client
- Classe *Requete*: à chaque top les requêtes recherchent un serveur cible (méthode *view_agt* du sous-domaine des clients correspondants). Pour cela, elles vérifient que le client n'a pas changé de sous-domaine depuis le top précédent, pour être sûres de chercher dans le bon sous-domaine. Chacune d'elles se dirige (méthode *aller_vers*) vers le serveur cible trouvé, la variable *distance_cour* mesurant la distance restante à parcourir. À son arrivée sur le serveur cible (distance_cour<seuil), elle vérifie s'il a déjà été visité (en fonction du protocole choisi). Si l'interaction est possible elle tente un calcul (méthode *lancer_calcul* du serveur cible) avec lui. Si le serveur est en capacité de faire le calcul (il possède suffisamment de ressources pour satisfaire la demande de la requête), le client est prévenu et la requête disparaît (mission accomplie). En cas d'impossibilité du serveur cible, ce dernier est ajouté à la liste des serveurs visités dans les données de la requête, et le client est prévenu (il augmente son nombre d'échecs). La requête vérifie son âge qui Requete

doit être inférieure à une durée limite (constante *AGE_MAX*), pour pouvoir continuer sa recherche au top suivant. Cela permet d'éviter des requêtes en errance, car incapables de trouver un serveur.

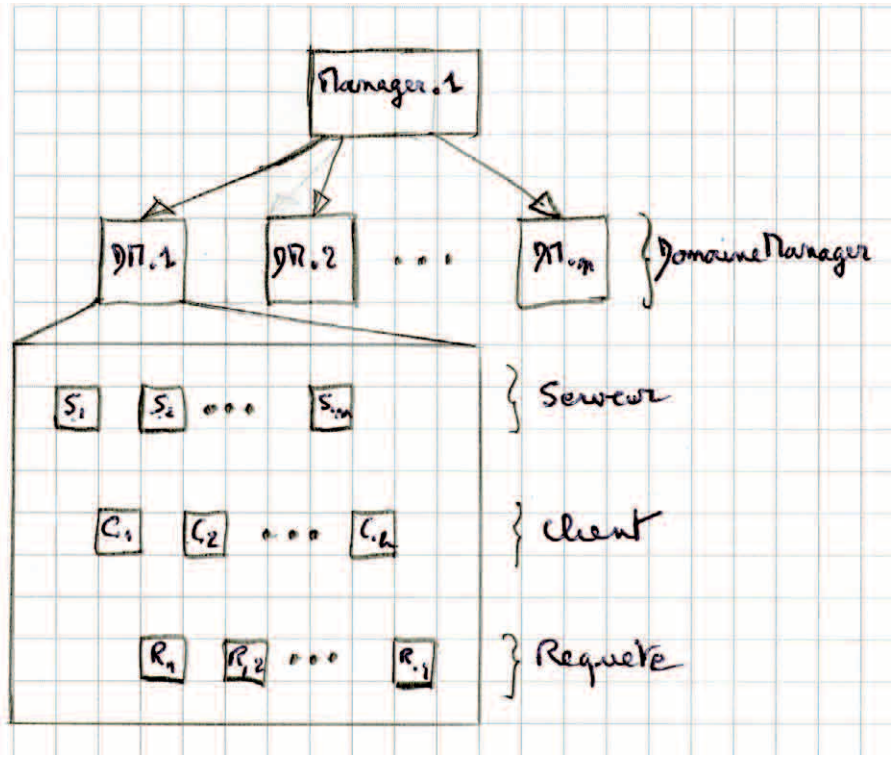


Fig-IV-D1-35

Le code des expérimentations

Chaque expérimentation demande de préciser les conditions initiales (combien de clients et de serveurs, leur position (x/y) dans l'environnement,…).
Il faudra également préciser les graphiques à afficher pendant l'exécution de l'expérimentation.
En utilisant la fonction *execute* du moteur *oRis*, et après avoir inclus «la simulation» (fichier «fusionV3.ors»), il y aura instanciation d'un *Manager*, puis d'un premier *DomaineManager*. Ce dernier créera les serveurs et les clients de l'expérimentation. Dès lors que les objets actifs seront instanciés, ils deviendront autonomes et l'expérimentation pourra démarrer.

IV-D2) Compléments algorithmiques et implantation

Le fonctionnement de la simulation, comme nous avons pu le voir, est quasiment décentralisé. La visualisation en temps réel de paramètres des objets actifs nécessite une centralisation temps réel de ces derniers sur le *Manager*. Par exemple, la variable *nbT_req* du *Manager*, correspond, dans une période d'observation (PO) donnée, au nombre total de requêtes ayant réussi à atteindre un serveur qui a traité favorablement leur demande. Cette variable est remise à zéro en début de chaque nouvelle PO. Cette variable est centralisée dans le *Manager* puisqu'elle englobe les requêtes de tous les clients du domaine.

Certains objets actifs (*Manager* et *Serveur*) fonctionnent sur la base de la notion de période d'observation (PO). Une PO se compose d'un nombre de tops d'horloge du moteur *oRis*, qui peut varier, dans le cas des serveurs, sous certaines configurations. Dans ce cas un traitement particulier sera fait dans la méthode *main* suivant que le top d'horloge courant est le dernier de la PO ou non. Le Manager ne fera rien pendant le dernier top de la PO, permettant ainsi aux autres objets actifs de la simulation, de lui mettre à jour des informations dans ses données propres. Pour les autres tops il mettra à jour ses données internes en vue d'un affichage temps réel sur ce qui se passe dans la simulation.
Malgré une décentralisation des décisions, la simulation utilise quand même, pour fonctionner, quelques mécanismes centralisés qui pourraient être supprimés. C'est le cas, par exemple, du recours aux *DomaineManager* dans la gestion (création/destruction) des clients et des serveurs.

Comment se passer du DomaineManager ?

L'idée du *DomaineManager* ou sous-domaine est de cloisonner le domaine (l'ensemble des clients et des serveurs) en sous-domaines afin de résoudre l'optimalité du système à l'intérieur de chaque sous-domaine, sans introduire des perturbations inter-domaines lors de processus de réorganisation locaux. Cette approche est particulièrement intéressante pour l'algorithme de division cellulaire, qui cherche à diviser le problème en sous problèmes.
L'alternative à une gestion par la classe *DomaineManager*, de la liste des clients (variable *Client[]Clvus) du sous-domaine*, consisterait, pour les serveurs, à construire dynamiquement l'enveloppe convexe du sous-domaine courant. Pour cela les serveurs utiliseraient les positions fixes des clients obtenues lors de leurs échanges avec les requêtes. En effet les requêtes possèdent les caractéristiques de leur clients géniteur qu'elles pourront partager avec les serveurs rencontrés. Lors des interactions indirectes (une requête visite une succession de serveurs en échec), les serveurs peuvent récupérer les coordonnées des autres serveurs rencontrés par les requêtes. Ils pourront échanger leurs connaissances sur les clients du sous-domaine et ainsi co-construire l'enveloppe convexe délimitant le sous-domaine correspondant.

Gestion des flux entrants sur les serveurs

Pour piloter les algorithmes de déplacement des serveurs, c.-à-d. recalculer, en fin de PO, de nouvelles directions à prendre vers la configuration optimale pour tous les serveurs, chacun d'entre eux mémorise les interactions locales qu'il entretient avec les clients et ses collègues lors de situations d'échec.
Pour cela chaque serveur dispose d'une liste (dynamique) d'objets *Interaction* (*Interaction[] interaction_srv*) qui sera mise à jour (méthode *maj_interaction*) au fur et à mesure que des requêtes arriveront sur lui. Un objet *Interaction* modélise l'interaction entre un client et un

serveur (notion de canal), via une requête et contient des informations sur cette interaction, mises à jour pendant la simulation. C'est le cas, par exemple, de la valeur cumulée des demandes (et/ou des échecs) traitées par ce canal pendant la PO courante. Une interaction appartient donc à un serveur. On y trouve le nom du serveur propriétaire, celui du client qui a provoqué cette interaction et éventuellement le nom du serveur visité juste avant le serveur courant, dans le cas des interactions indirectes. Nous verrons comment la méthode *restructurer* utilise la variable *cpt_explo_etats*, qui est incrémentée à chaque arrivée d'une requête via un serveur (interaction indirecte), uniquement si le client n'est pas nouveau. Cette variable est remise à zéro dès qu'un nouveau client arrive sur le serveur, via une requête.
Cette liste, mise à jour en temps réel, est utilisée par le serveur dans la recherche de sa direction à prendre (méthode *restructurer*).
En complément à cette liste, chaque serveur calcule l'enveloppe rectangulaire de recouvrement de son sous-domaine (variables de classe *xdom*, *ydom*, *Xdom* et *Ydom*) et mémorise dans la liste «*Client[] cls_dom*» la liste des clients de son sous-domaine.

Dans la méthode *main* de la classe *Requete*, quand une requête arrive à proximité du serveur cible («if(distance_cour<size/5)»), elle tente une interaction avec ce serveur («etat_calcul=srv_cour->lancer_calcul(demande,mon_client,srv_visite)»).
Le serveur visé regarde si le calcul est faisable (si la demande de la requête est inférieure aux réserves du serveur). En cas de succés le serveur met à jour:

- ses réserves;
- l'interaction correspondante (méthode *maj_interaction*);
- et chez le client de départ, la distance euclidienne entre lui et ce dernier.

La mise à jour de l'interaction consiste soit à ajouter un nouvel objet *Interaction* dans la liste des interactions, si le client est nouveau pour ce serveur, soit à mettre à jour l'objet *Interaction* correspondant au client visé.
Il y a deux possibilités en fonction de l'origine de la requête. Soit elle vient directement d'un client (son champ «dernier serveur visité» est à NULL), soit elle a subi un échec sur un serveur différent du serveur courant. Dans le fonctionnement de base des heuristiques, l'échec d'une requête sur un serveur, lui interdit de retenter une interaction immédiate sur le même serveur. Elle pourra revenir après avoir été sur un autre serveur, afin d'éviter des situations de bouclage.
Ensuite il faudra refaire le calcul de l'enveloppe rectangulaire de recouvrement du sous-domaine (mise à jour des variables *xdom/ydom* et *Xdom/Ydom*). Ce calcul n'est pas nécessaire si la requête est passée par un autre serveur. En effet les clients étant fixes, lors des échanges entre ces serveurs, une mise à jour du rectangle de recouvrement aura été faite. Il faut ensuite ajouter ce nouveau client dans la liste des clients locaux au serveur (*cls_dom*). Si le mode expansion de l'algorithme de restructuration est en cours, alors on arrête le déplacement du serveur courant.

Les serveurs échangent également, entre eux, les listes des clients sur le même principe que celui utilisé pour les rectangles de recouvrement.

L'ALGORITHME DE RESTRUCTURATION DES SERVEURS

Cet algorithme est lancé par chaque serveur, en fin de période d'observation (PO), pour recalculer une nouvelle direction de déplacement pour la PO suivante. Il est composé de trois stratégies différentes «marcheur aléatoire», «descente de gradient» et «contraction/expansion»:

- l'algorithme commence par regarder auprès du *Manager* si la simulation est configurée en mode «marcheur aléatoire» (constante *MARCHE_ALEA* à *true*). Dans ce cas le serveur se met en mode «marcheur aléatoire». En s'appuyant sur les objets *Interaction* (incluant les clients qu'il voit), il explore, en mode aléatoire, l'enveloppe de recouvrement de son sous-domaine courant. Cette enveloppe délimite la zone de l'espace d'états à explorer que le serveur discrétise (grille de points) pour être plus efficace dans son choix de direction à prendre. Nous rappelons qu'un état correspond à une configuration/répartition donnée entre les clients et les serveurs du sous-domaine concerné. La constante *EXPLO_CL_ETATS* (positionnée dans le fichier «fusionV3.cstes») détermine le nombre d'échanges consécutifs (via les requêtes en échec-interaction indirecte) entre deux serveurs pour lesquels la liste des clients vus (variable *cls_dom*) du domaine ne varie pas. Ce seuil permet de choisir au bout de combien d'interactions (variable *cpt_explo_etats*) on estime qu'une situation est suffisamment stable (ici dans l'échec) pour que le serveur courant passe de la découverte des clients du sous-domaine, à une exploration de ses états. Si ce seuil n'est pas atteint, on choisit aléatoirement l'un des clients vus par le serveur;
- si le mode «descente de gradient» a été positionné, on vérifie juste qu'il existe bien au moins un objet *Interaction*. Il n'y a rien d'autre à faire, car dans ce cas, ce sera à chaque top d'horloge, dans le *main* du serveur cible que le choix d'une direction sera faite en fonction des possibilités données par le gradient. S'il n'existe aucun objet *Interaction*, alors on arrête le déplacement du serveur;
- dans ce troisième cas, chaque serveur met en œuvre l'algorithme appelé «contraction/expansion» (cf. la figure *Fig-IV-D1-36*). Grâce à ses objets de la classe *Interaction*, il va d'abord identifier, parmi ses clients, celui qui a la plus forte demande, qu'il soit lui-même en échec ou non. S'il est en échec, il regarde si son pourcentage de demandes non satisfaites dépasse le seuil de contraction, fixé par la constante SEUIL_CONTRACT (à 5% dans les simulations) dans le fichier «fusionV3.cstes». Dans ce cas il part en contraction, c.-à-d. se déplace vers son client de plus forte demande. Sinon, il

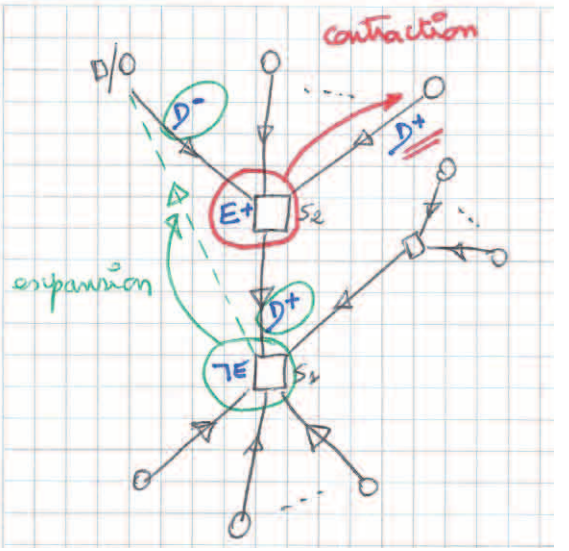


Fig-IV-D1-36

n'a aucun, voire peu, d'échecs, alors il tente une expansion vers une cible qu'il devra calculer. Pour cela il regarde parmi les serveurs qu'il voit, celui qui possède un client (ou un serveur) de plus forte demande non satisfaite. Dans ce cas il se dirige vers cette cible. Il s'agit de profiter d'une production suffisante, du serveur courant, pour aller capturer des demandes insatisfaites de clients ayant généré des échecs sur des serveurs vus par le serveur courant. Pour cela le serveur courant doit se rapprocher de la nouvelle cible, sans perdre ses clients tout en capturant de nouveaux. Concrètement, en situation d'expansion (peu d'échecs), le serveur va aller vers l'élément précédant (dans la chaîne d'interaction construite via les requêtes) le serveur qu'il voit et qui se trouve dans la situation de plus fort échec (en volume). Cette expansion durera tant qu'il n'y aura pas suffisamment d'échecs (à la limite du seuil).

Il suffit de modifier le code de la méthode *restructurer* pour mettre en place de nouvelles stratégies de réorganisation. Dans tous les cas il faut garder à l'esprit que le calcul d'une nouvelle stratégie se fait sur le dernier top de chaque période d'observation (PO), alors que le calcul d'un déplacement est réalisé potentiellement à chaque top d'une PO (sauf le dernier qui assure la cohérence de l'affichage des résultats en temps réel).

La stabilité des serveurs

La notion de stabilité d'un serveur est très importante dans la recherche de la solution optimale au problème global posé par C_iS_j. Les serveurs auto-ajustent leur comportement dans des processus décentralisés, afin de trouver collectivement la meilleure solution. Pour cela ils doivent être en capacité de mesurer à tout instant leur propre stabilité. Dans l'espace des phases d'un système dynamique classique, il existe des attracteurs dont certains correspondent à des minima locaux. Il est crucial pour le système de différencier les attracteurs locaux des attracteurs optimaux recherchés. Le système doit pouvoir mesurer la forme de la stabilité qu'il rencontre dans ces attracteurs locaux, afin de pouvoir en sortir.
Dans notre situation, la mobilité des serveurs joue un rôle crucial dans l'instabilité du système. En effet il suffit qu'un serveur se déplace pour provoquer une rupture d'équilibre sur les serveurs et les clients concernés. Nous devons définir cette notion de stabilité des serveurs.
Les études théoriques de la résolution de C_iS_j, vues au chapitre III, montrent qu'une découpe du problème en régions permet de le résoudre par parties, tout en cassant la complexité d'une résolution monobloc. Nous verrons dans les expérimentations comment cette décomposition du problème, en parties disjointes, nécessite des ajustements locaux pour atteindre les fameux minimum locaux, voire des réorganisations plus larges pour remettre en cause le partitionnement en cours du système sans rompre l'équilibre atteint. Plusieurs méthodes permettent la réalisation d'ajustements locaux (descente de gradient, calcul exact

sur une grille approchée aux dimensions réalistes,…). L'essentiel repose sur le fait que les serveurs doivent savoir s'ils sont stables, à partir de données dont ils disposent ou qu'ils peuvent se procurer.
Nous avons introduit la notion de niveau de stabilité pour un serveur donné. Au premier niveau, le serveur stable ne se déplace plus, n'a plus ou peu d'échecs induits et sa production de ressources correspond exactement à la quantité demandée par les clients, via les requêtes. Cette stabilité est plus au moins robuste dans le temps, car, malgré la coopération entre serveurs, il suffit que des clients aient des besoins hiératiques, pour que des situations de perturbations cycliques interviennent et rendent très difficile cette stabilité.
À priori un serveur stable ne se déplace plus. Cette condition nécessaire n'est pas toujours suffisante, car on peut être amené à considérer la stabilité des serveurs qui coopèrent.

Dans nos simulations, la méthode *stable* permet aux serveurs de mesurer leur niveau de stabilité. Elle renvoie 0 en cas d'instabilité du serveur courant, 1 si sa stabilité est considérée comme temporaire, 2 si elle est localement forte et 3 si le serveur courant est dans un attracteur. Le niveau 3 nécessite la stabilité forte des serveurs voisins (en coopération avec lui). Pour cela la méthode *collegues_stable* regarde si au moins l'un des «voisins», du serveur courant, est stable au niveau 2. Ceci n'est pas entièrement satisfaisant et mérite une étude expérimentale approfondie.
Dans nos heuristiques, lorsqu'un serveur se sait dans une stabilité de niveau 3, il peut soit lancer un calcul exact sur la grille (discrétisation de la zone où il se trouve) pour connaître les différentes valeurs des coûts associés (CG) vers l'attracteur local, soit lancer une division cellulaire (mitose) en se dédoublant.

Un serveur, dont sa stabilité est de niveau 2, peut lancer le calcul de la grille (s'il est accessible), alors qu'au niveau 1, sa stabilité temporaire lui permet de lancer le mécanisme de propagation de la valeur de sa période d'observation (PO). L'idée sous-jacente est appuyée par notre étude théorique (cf. le chapitre III). Elle consiste à dire que l'optimal absolu du système global nécessite le partage d'une valeur de PO entre les serveurs, valeur qui dépend du contexte (de la nature du problème).
L'idée est donc de propager la PO d'un serveur à ses «voisins», lorsqu'il a atteint un certain niveau de stabilité. Ces échanges permettent de synchroniser ces serveurs grâce à des valeurs de PO communes. Nous verrons avec les expérimentations les possibilités de cette approche.

La méthode *agir_stable* met en œuvre cette propagation de PO. Nous sommes donc dans le cas où le niveau de stabilité du serveur est temporaire, mais suffisant pour permettre le calcul (si faisable en temps machine) des positions optimales des serveurs, correspondant à l'état attracteur recherché.

La première étape consiste, pour le serveur courant, à récupérer la liste de ses clients entrants et leurs coordonnées, afin de construire le rectangle englobant de cet ensemble de clients (Xmin/Ymin, Xmax/Ymax). Le serveur lance alors, via la méthode *calcule_envConv*, le calcul approximatif (pessimiste et rapide) du nombre de points sur la grille recouvrant ce rectangle englobant.
Chaque point retenu nécessitant un calcul de coût (somme des distances de chaque client à ce point), le serveur vérifie que l'ensemble des calculs à faire ne dépasse pas un certain seuil (constante *SEUIL_CALCULABILITE* du fichier «fusionV3.cstes»). Il faut bien comprendre que ces calculs devront être faits pendant l'exécution de la simulation, c.-à-d. au cours d'un top d'horloge du moteur d'*oRis*.
Sauf à utiliser le mode hautement parallèle, que nous n'utilisons pas, a priori, exécuter une tâche très longue dans la boucle d'un objet actif, revient à ralentir toutes les boucles des autres objets actifs de la simulation… Ce seuil est donc fixé au niveau des constantes globales de la simulation. On pourrait le rendre dépendant d'objets actifs qui représenteraient les nœuds de calcul du réseau, avec des puissances de calcul variables. Nous n'avons pas choisi cette approche pour ne pas compliquer la simulation et ainsi passer à côté des phénomènes qui nous intéressent. Cependant nous pourrions le faire sur *oRis*. Il suffirait de créer une classe *nœud* et travailler dans un environnement complètement discrétisé (graphe). On peut renvoyer le lecteur à l'étude théorique, au chapitre III, pour avoir des éléments précis sur les limitations rencontrées.
Le calcul des points à conserver, via le rectangle englobant des clients (méthode *calcule_envConv*), pourrait être optimisé en ne gardant que les points du rectangle qui appartiennent à l'enveloppe convexe des positions des clients. Le calcul de l'enveloppe convexe ajouterait du temps aux calculs, mais en ferait gagner sur le nombre de calculs à faire… Si le seuil de calculabilité n'est pas atteint, les calculs sont lancés via la méthode *calcule_grille*.
Les variables *TAILLEbis_x/TAILLEbis_y* servent à piloter la double boucle (*i/j*) de cette méthode *calcule_grille*, dont le but est de parcourir tous les points de la grille pour effectuer le calcul des CG. Sachant que la grille se définit aussi par sa *MAILLE*, qui permet de calculer le nombre de points entre deux entiers (*[1/MAILLE]-1), nous avons les valeurs suivantes*:

- si MAILLE=1 on a zéro point entre deux entiers;
- si MAILLE=½ on a un point entre deux entiers;
- si MAILLE=¼ on a deux points entre deux entiers
- …

Il s'agit donc de discrétiser l'espace à partir des positions des clients qui sont continues (réels). Si les positions des clients sont entières, le calcul des points retenus de la grille fonctionne avec la *MAILLE* sinon cela se complique.
Une fois les calculs effectués, les résultats sont stockés dans un tableau, en partant de la plus faible valeur de CG. Il suffit de prendre cette valeur pour connaître la direction à prendre.

Si le calcul de l'optimal n'avait pas été possible, la méthode *agir_stable* regarde si les conditions sont réunies pour faire une division cellulaire du serveur en appelant la méthode *fork_srv*. Via cette méthode, le serveur demande à son *Manager* de mettre en place un sous-domaine (*DomaineManager*) à l'intérieur de son propre sous-domaine et d'y placer le serveur cloné. Cela protège les autres régions en équilibre, des réorganisations dues à l'arrivée de ce nouveau serveur. L'idée est de mettre le serveur courant et ses clients dans un nouveau sous-domaine avec le nouveau serveur dupliqué. La région de départ sera ainsi divisée en deux régions (mitose), une par serveur. De cette façon le nombre de serveurs, dans la résolution générale, pourra croître jusqu'à atteindre l'optimal recherché !

Lorsqu'un serveur est instable, il récupère la valeur de la période d'observation (PO) d'un voisin stable. Une variante consiste à récupérer les valeurs des PO de plusieurs voisins stables et d'en calculer la moyenne. C'est ici que l'on peut améliorer le processus de convergence vers une PO optimale, en travaillant sur la coopération entre serveurs voisins.

Le calcul de la matrice des états

Le calcul de la matrice des états du système (C_iS_j) est réalisé au fur et à mesure de la résolution, grâce à la méthode *calcul_matrice_etats* de la classe *DomaineManager*. Chaque sous-domaine calcule donc en permanence (à chaque top de la simulation) sa propre matrice. Comme vu précédemment, une matrice correspond en réalité à un graphe. Elle se compose de nœuds (les états) et d'arcs entre ces nœuds. Le passage du système d'un état E_1 à un état E_2 crée un arc entre ces deux états, ou, s'il existe déjà, renforce la valeur du lien associé à cet arc. Cette matrice nous informe du nombre d'états du système, mais aussi sur la façon dont le système évolue dans l'espace de ses propres états (liens entre eux et valuation de ces liens).

L'objectif de la méthode *calcul_matrice_etats*, étant de mettre à jour la matrice des états, il lui faut créer les composants correspondant aux nouveaux états rencontrés lors de l'exploration/résolution.

Le paramètre «int[]un_etat» est un tableau dynamique[1] d'entiers, où chaque valeur entière représente l'indice du serveur vu par le client correspondant. «Serveur[]srv_mes_clts» est également un tableau dynamique où chaque élément représente l'un des serveurs vus (en interaction) avec l'un des clients du sous-domaine. Au final, ce tableau contient tous les serveurs vus par les clients du sous-domaine.
La variable *nb_clts* est le nombre de clients du sous-domaine, alors que *serv_vu* est le serveur le plus proche du client courant.

1 Dans *oRis*, la notion de tableau dynamique permet d'ajouter (méthode *insert*) au fur et à mesure des éléments au tableau sans préjugé sur la longueur finale du tableau. C'est l'équivalent d'un *malloc/realloc* en 'C'.

Techniquement cette méthode fonctionne en trois phases 'a', 'b' et 'c' :

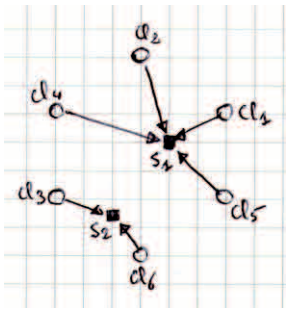


Fig-IV-D2-37

- a) cette première phase consiste, pour chaque client (*Cl_1, Cl_2, Cl_3, Cl_4, Cl_5 et Cl_6* sur l'exemple de la figure *Fig-IV-D2-37*) à chercher son serveur le plus proche (*serv_vu*). Chaque serveur trouvé est inséré dans *srv_mes_clts*, construisant ainsi la table des serveurs vus (srv_vu={S_1,S_2} sur la figure *Fig-IV-D2-37*) par tous les clients du sous-domaine. On va ensuite remplir le tableau dynamique *un_etat* («(un_etat).insert(i,1);»),qui sert pour le calcul de l'entier représentant l'état trouvé. Sur l'exemple de la figure *Fig-IV-D2-37*, nous avons un_etat={1,1,2,1,1,2}. Cela correspond au fait que les clients Cl_1, Cl_2, Cl_4 et Cl_5 sont proches du serveur S_1, alors que les clients Cl_3 et Cl_6 sont proches du serveur S_2;
- b) cette deuxième phase consiste, en partant du tableau d'entiers *un_etat*, à calculer l'état trouvé sous la forme d'un entier, et le mettre dans une variable locale («int etat=0;»), en respectant la contrainte de ne pas avoir deux entiers distincts qui pointent des états identiques. Plusieurs techniques peuvent être utilisées. Nous en avons choisi deux, l'une avec les puissances de 10 et l'autre sous la forme d'un entier de 32 à 63 avec les puissances de 2;

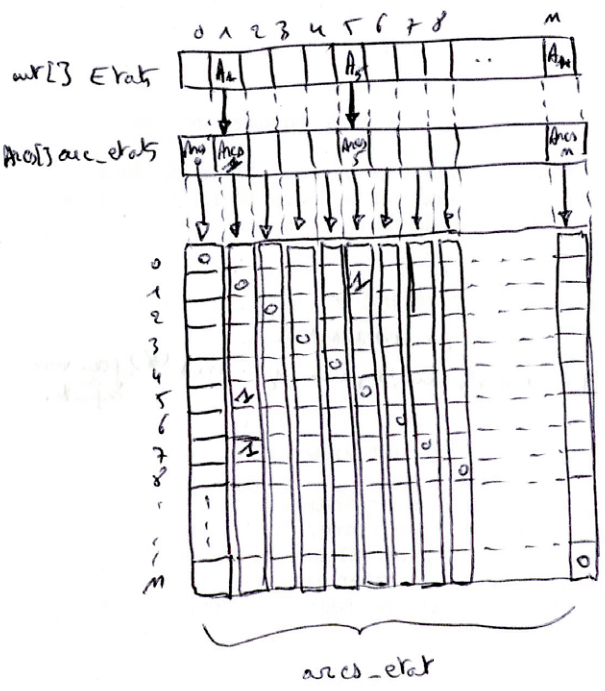


Fig-IV-D2-38

- c) enfin la dernière phase de la méthode *calcul_matrice_etats* vérifie si cet état courant est nouveau ou non dans la matrice déjà construite du sous-domaine courant. Pour cela le *DomaineManage* courant dispose du tableau dynamique d'entiers «int[] Etats;» qui est le vecteur des états trouvés. Il dispose également du tableau dynamique «Arcs[]arc_etats;» qui représente les arcs (objets *Arcs*) entre les états. Chaque objet *Arcs* possède une variable locale «int[] arcs_etat;» qui est un tableau dynamique d'entiers de valeur 1 ou 0, en fonction de la connexion ou non avec un autre état de la matrice. On parcourt *Etats*. Si l'état que l'on vient de calculer existe déjà dans l'ensemble des états («if(etat==Etats[i]) j=i;»), il suffit de mettre à jour l'arc de l'état précédent comme pointant vers l'état courant («arc_etats[indice_prec]->arcs_etat[j]=(arc_etats[indice_prec]->arcs_etat[j])+1;»). Si cet état courant n'existe pas dans les états connus, il faut créer un nouvel état dans *Etats* («(Etats).insert(i,etat);»), mais aussi insérer un nouvel arc dans la liste *arc_etats* («(arc_etats).insert(i,arcs_cour);»).

Le poids associé à chaque arc_{ij} (de l'état *Etats[i]* vers l'état *Etats[j]* et la réciproque) est calculé à partir de la matrice, par la méthode *sGS* du *DomaineManager* correspondant, au moment de la génération du fichier «GS_Etat_DomaineManager.k.dgs» à destination de *GraphStream* (logiciel de visualisation interactif de graphes). Il correspond au nombre de fois que cet arc aura été traversé

pendant la résolution. Il apparaît sous la forme d'un label sur l'arc dans le graphe des états.

IV-D3) Les outils complémentaires de visualisation

À cela il faut ajouter les outils complémentaires que nous avons du développer pour compléter nos études. *oRis* offre de nombreuses possibilités, comme nous le verrons sur un exemple, pour afficher en temps réel des courbes, en plus d'offrir en temps réel une vision 2D/3D de l'environnement d'exécution des objets actifs.

Un générateur de grille

Dans notre étude particulière autour de l'alphabet nous avons créé un script *perl* («genererereseau.pl») en amont de la simulation afin de construire une grille qui s'intégre dans l'environnement des expérimentations sous *oRis*. Cette grille permet de mieux comparer les expérimentations sur l'alphabet avec les études théoriques du chapitre III. Les autres expérimentations, hors alphabet, ne nécessitent pas cette grille.

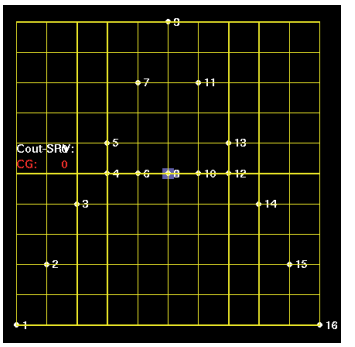


Fig-IV-D3-39

```
genererereseau.pl 11 11 initGRILLE.ors
```

En lançant le script *perl* avec les paramètres ci-dessus, on obtient dans le fichier «initGRILLE.ors», une grille avec cent-vingt-et-un (11x11) points («Node n25=NEW Node(2,3);»), cent-dix (10x11) segments horizontaux («NEW Link(0.01,4,n12,n23);») et autant de segments verticaux.
En incluant ce fichier («include "initGRILLE.ors"») dans «fusionV3.ors», on obtient le tracé de la grille jaune que l'on peut voir sur la figure *Fig-IV-D3-39*.

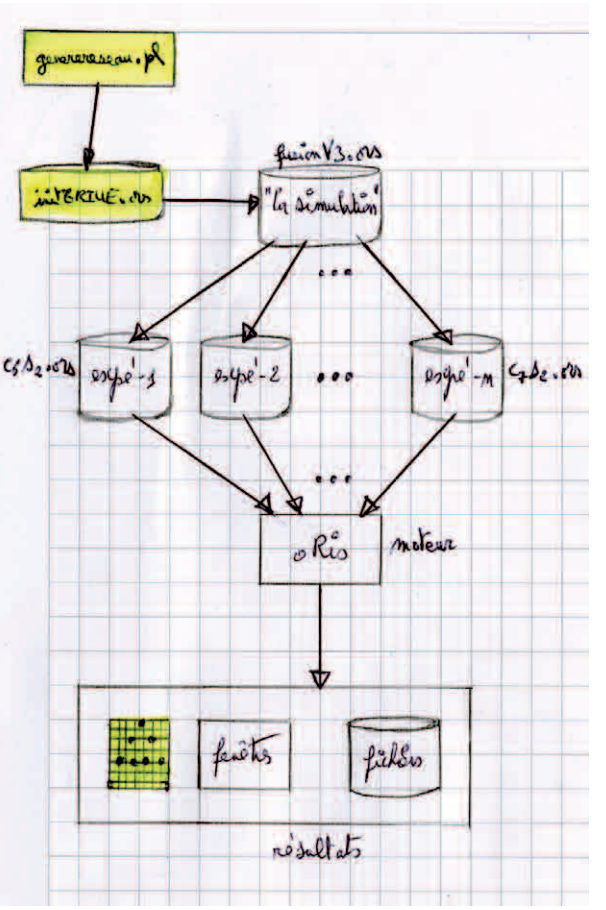


Fig-IV-D3-40

Un générateur de graphes d'états interactifs

Un état dans C_iS_j est défini conjointement par le positionnement des 'j' serveurs dans l'environnement (sous-domaine correspondant), avec la répartition correspondante des 'i' clients sur ces derniers.

L'idée de faire un graphe de ces états est de comprendre la dynamique du système de résolution. Chaque nœud du graphe représente un état. Un arc entre deux états représente la possibilité, pour le système, de passer d'un état à un autre.

En étudiant les caractéristiques de tels graphes (densité, profondeur moyenne,…), on peut comprendre la complexité de la résolution et adapter les heuristiques en conséquence.

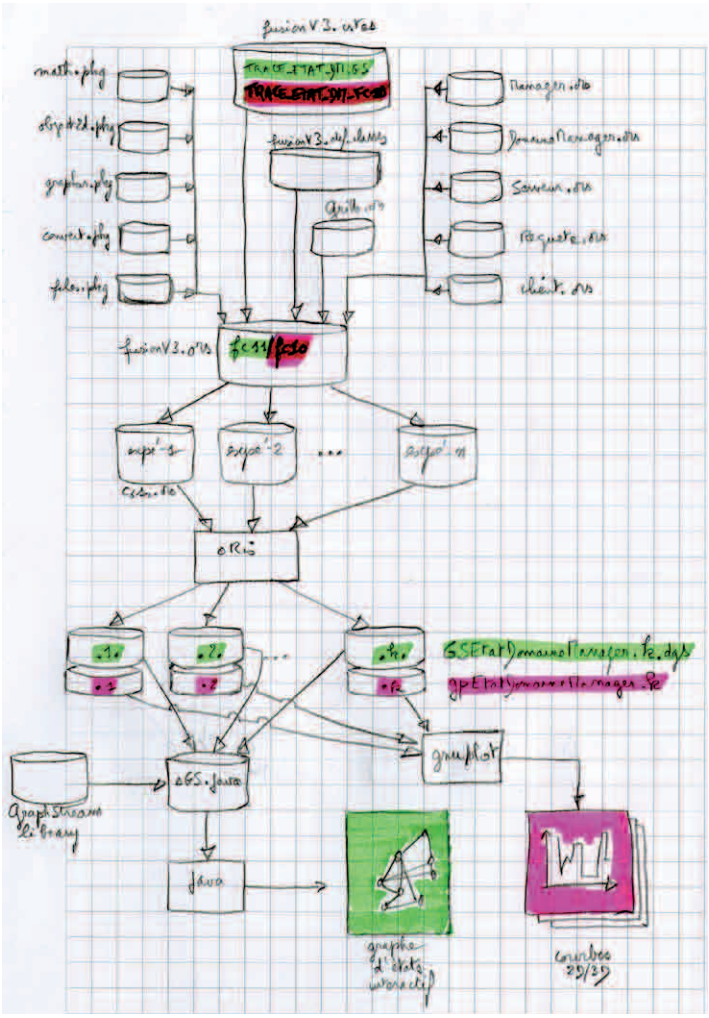


Fig-IV-D3-41

La simulation stocke (méthode *calcul_matrice_etats* des *DomaineManager*) les états dans des données (matrice,…), au fur et à mesure que la résolution avance (exploration).

Comme le montre la figure *Fig-IV-D3-41*, l'affichage des différents graphes d'états (un par sous-domaine) nécessite en premier lieu de positionner à *true*, dans le fichiers «fusionV3.cstes», les constantes *TRACE_ETAT_DM_GS* (pour l'affichage au format *GraphStream* via les fichiers «GSEtatDomaineManager.k.dgs») et *TRACE_ETAT_DM_FC10* (pour la sauvegarde des états dans un fichier texte exploitable par *gnuplot*).

Pour pouvoir être exploitées par *GraphStream*, les données générées devront respecter certaines contraintes du visualisateur de graphe. Un nœud ne peut être en relation avec lui-même et les arcs inverses ne fonctionnent pas non plus à l'affichage. Ils seront donc affichés en rouge dans *GraphStream*. Chaque arc ne pouvant être valué, il le sera via la notion de *label*. Un *label* correspond au nombre de fois qu'un arc a été franchi pendant la résolution (cf. les valeurs sur les arcs de la figure *Fig-IV-D3-42*). Cette valeur cumule les passages dans un sens avec ceux dans l'autre.

java sGS

En lançant le fichier «sGS.java» sur la machine virtuelle *java*, la lecture du fichier «GSEtatDomaineManager.1.dgs» fournit les informations aux librairies *GraphStream* pour construire et afficher le graphe des états interactifs. Les nœuds du graphe peuvent être déplacés à la souris sans perdre les arcs qui les relient entre eux.

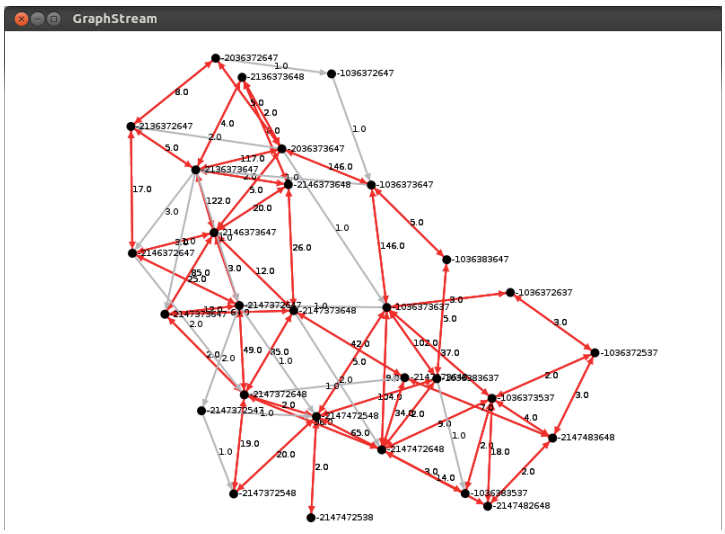


Fig-IV-D3-42

Sur l'exemple de la figure *Fig-IV-D3-42*, nous avons un graphe généré par *GraphStream* à partir d'une simulation. Ce dernier contient vingt-neuf états et cent-vingt-trois arcs. Certains arcs n'ont été traversés qu'une seule fois, alors que d'autres l'auront été cent-quarante-six fois.

Avant de lancer une nouvelle simulation, il faut détruire les fichiers de sortie des états, car

l'écriture est réalisée en mode concaténation (*append*) :

```
rm GSEtatDomaineManager.?.dgs
```

Une fois la simulation stabilisée et avant de sortir d'*oRis*, il faut lancer manuellement l'écriture des graphes d'états par sous-domaine. Pour cela dans *oRis*, faire :

```
::execute
{
Manager.1->sGSALL();
DomaineManager.1->fc11->close();
}
```

La méthode *calcul_matrice_etats* de la classe *Domaine-Manager*, actualise, si la constante *TRACE_ETAT_DM_GS* est à vrai, à chaque top du moteur *oRis* la matrice des états, mais également, si la constante *TRACE_ETAT_DM_FC10* est à *true*, l'évolution des états dans le fichier «gpEtatDomaineManager.k» (via le canal fc10).
On trouve, sur chaque ligne de ce fichier des états, le nom de l'état (valeur entière calculée à partir des puissances de deux), la distance cumulée entre tous les clients et les serveurs associés du sous-domaine courant (DGoS), ainsi que le coût cumulé (produit de *distance* par *volume_de_la_demande*) entre tous les clients et les serveurs associés du sous-domaine courant (CGoS).

Dans l'exemple de la figure *Fig-IV-D2-37*, avec six clients et deux serveurs, nous avions un état E_i={1,1,2,1,1,2} dont la valeur entière associée serait :
etat=etat+pow(2,nb_clts-i-1)*un_etat[i], soit
2^5x1+2^4x1+2^3x2+2^2x1+2^1x1+2^0x2=72.

La figure *Fig-IV-D3-43* nous donne un aperçu du contenu du fichier «gpEtatDomaineManager.1» :

- la deuxième colonne affiche les valeurs entières des états traversés lors d'une résolution ;
- la troisième colonne affiche les valeurs des distances cumulées (DGoS) ;
- la quatrième colonne affiche celles des coûts cumulés (CGoS).

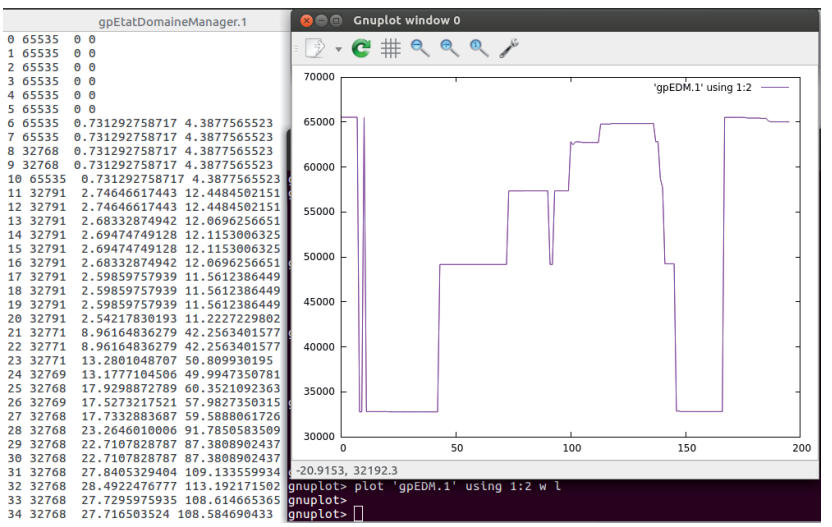


Fig-IV-D3-43

Sur la partie droite de cette même figure Fig-IV-D3-43 s'affiche la sortie *gnuplot* du fichier correspondant aux instructions suivantes, avec en abscisse la première colonne (n° des tops d'horloge du moteur *oRis*) et en ordonnée la deuxième (valeurs entières associées aux états) :

```
gnuplot> set surface
gnuplot> set contour both
gnuplot> set hidden3d
gnuplot> set dgrid3d 20,20,3
gnuplot> plot 'gpEtatDomaineManager.1' using 1:2
```

Cela nous montre comment la résolution explore chronologiquement l'espace des états du système C_iS_j. Le début de la courbe montre que la résolution est passée de l'état 65535 à l'état 32768, puis est revenue à l'état 65535, avant de repartir «durablement» à l'état 32768. Les paliers montrent comment la résolution cherche une solution à l'intérieur d'un état. La longueur du palier indique le temps passé à l'exploration à l'intérieur d'un état. Cette courbe montre d'éventuelles oscillations entre états, le nombre d'états en jeu,…

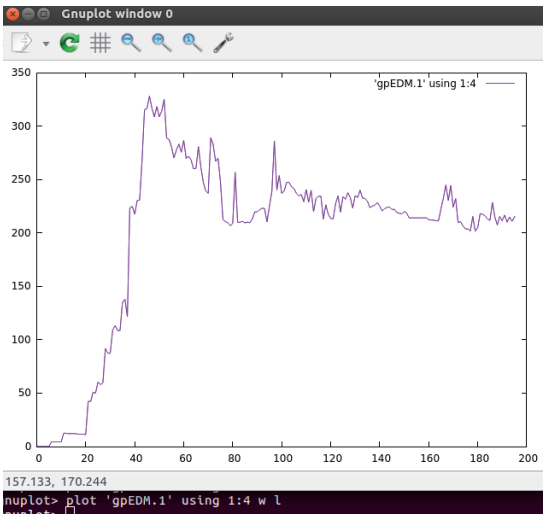


Fig-IV-D3-44

On peut voir à l'intérieur d'un état comment la résolution progresse par rapport à la mesure de la solution (ici CGoS). Par exemple entre le top n°11 et le top n°20 le système est dans l'état 32791. La valeur CGoS du coût global passe progressivement de 12,4484… à 11,2227…, sachant que la résolution cherche la meilleure solution (CGoS minimal).
La figure *Fig-IV-D3-44* montre sur le même exemple l'évolution du coût global (CGoS) en fonction du temps (les tops du moteur). Cette résolution a nécessité trente-quatre passages d'états. Il aura fallu environ quarante tops pour atteindre la demande de «croisière» (tous les clients ont des requêtes qui ont atteint leur serveur associé). On voit nettement ici que la résolution converge vers un minima. Tout le problème du réglage des heuristiques sera de faire converger le système vers l'optimum. Nous verrons cela dans la suite de cette partie.

Un générateur de partitions sous *oRis*

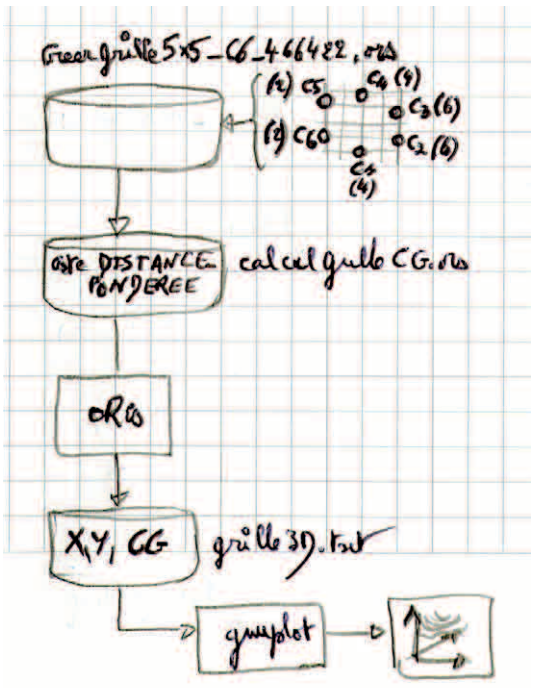


Fig-IV-D3-45

Cet outil, écrit sous *oRis*, permet de calculer en chaque point d'une grille la valeur du coût global (CG), si un serveur était positionné sur ce point. On récupère les résultats (positions x/y et le CG associé) dans le fichier «grille3D.txt» qui est exploité visuellement par *gnuplot*.
Le fichier «calculGrilleCG.ors» est lancé dans *oRis*. La constante *DISTANCE_PONDEREE* permet, si sa valeur est à «vrai» de tenir compte (pondération) dans le calcul du coût (CG) des demandes des clients.

Dans l'exemple de la figure *Fig-IV-D3-45*, nous avons six clients (C_1,C_2,C_3,C_4,C_5,C_6) dont les demandes en ressources sont respectivement quantifiées par 4, 6, 6, 4, 2 et 2.
Le fichier «CreerGrille5x5_C6_466422.ors» configure le programme en définissant la taille de la grille, ici vingt-cinq (5x5) points, le nombre, la position et la demande des clients. On peut également définir la *MAILLE* de la grille, pour avoir des points entre les unités.

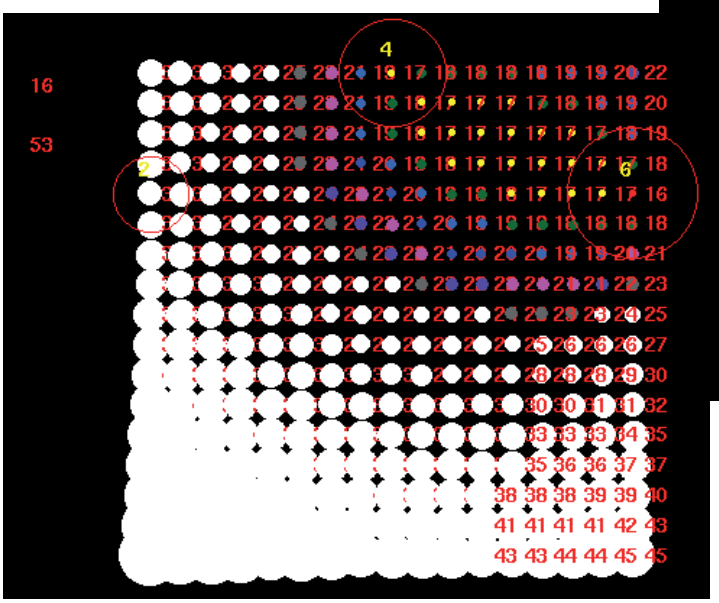
Dans l'exemple présenté avec les figures *Fig-IV-D3-46/49*, nous avons une grille de vingt-cinq (5x5) points également, mais avec une *MAILLE* de ¼. Cette notion permet, en gardant la même grille, mais avec plus de points, d'affiner la valeur des calculs.

Pour changer de configuration, et donc de grille, il suffit de prendre un autre fichier, comme par exemple le fichier «CreerGrille15x15_C21_cercle.ors» qui possède une grille de deux-cent-vingt-cinq (15x15) points, avec vingt-et-un clients répartis sur un cercle. Plus la grille est grande et plus les calculs seront longs.

Pour regarder une partition d'un système donné, il suffit de mettre en commentaire, au moment de la création des clients, dans le fichier «CreerGrille5x5_C6_466422.ors», ceux qui n'appartiennent pas à cette partition.

Les figures *Fig-IV-D3-46/49* montrent, dans *oRis*, les partitions incluant respectivement tous les six clients, les clients $\{C_2,C_3,C_4\}$, les clients $\{C_1,C_5,C_6\}$ et les clients $\{C_3,C_4,C_5\}$. On voit sur chaque point de la grille la valeur du coût (CG) pour les clients de la partition concernée. Un jeu de couleurs permet de repérer les points ayant les meilleurs valeurs (en rouge pour CG_{min}). Les valeurs CG_{min} et CG_{max} sont affichées en rouge en haut à gauche à l'extérieur de chaque grille.

En regard de chaque sortie, *oRis*, de la grille sur les figures *Fig-IV-D3-46/49*, nous avons, sur les figures *Fig-IV-D3-50/54*, les affichages équivalents (en trois dimensions) en utilisant *gnuplot* avec le fichier «grille3D_GP.txt».



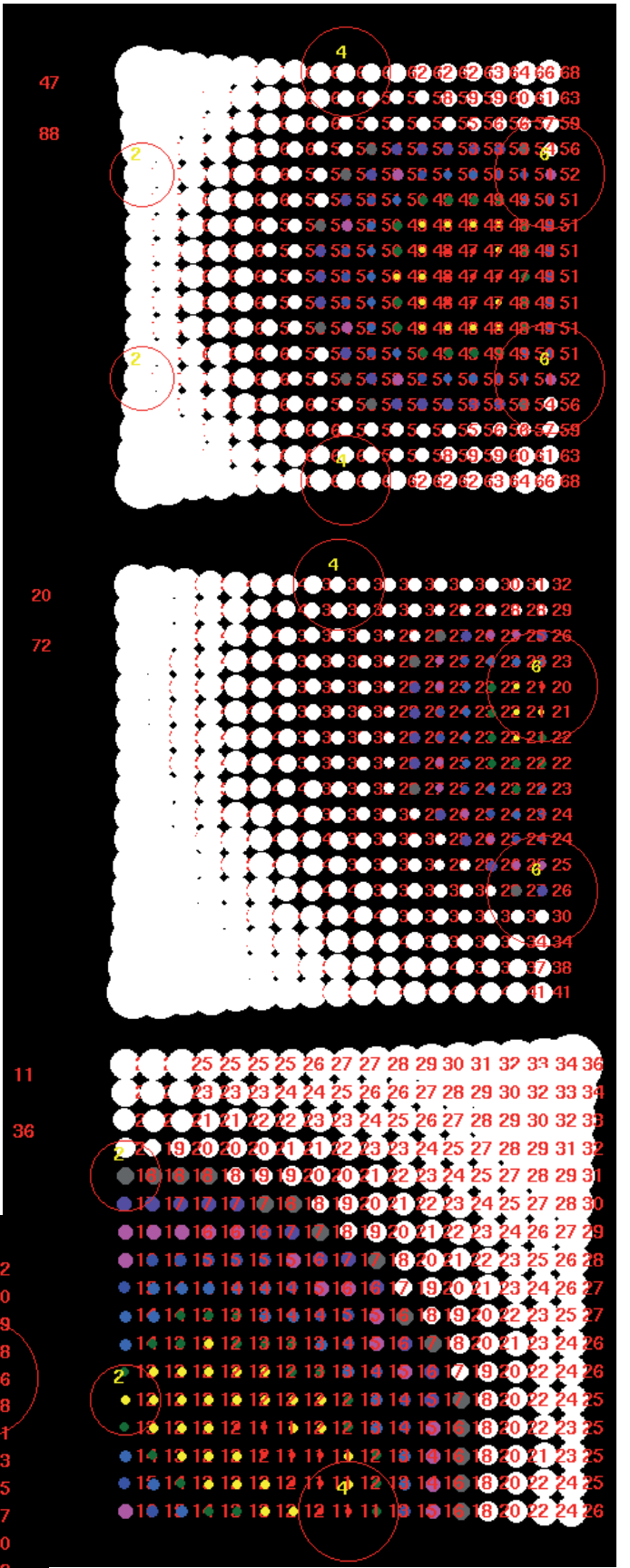
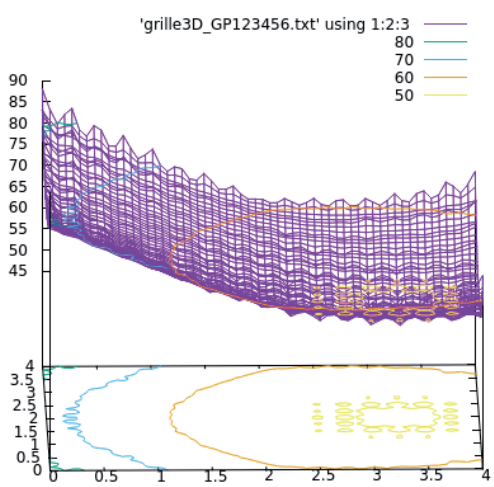
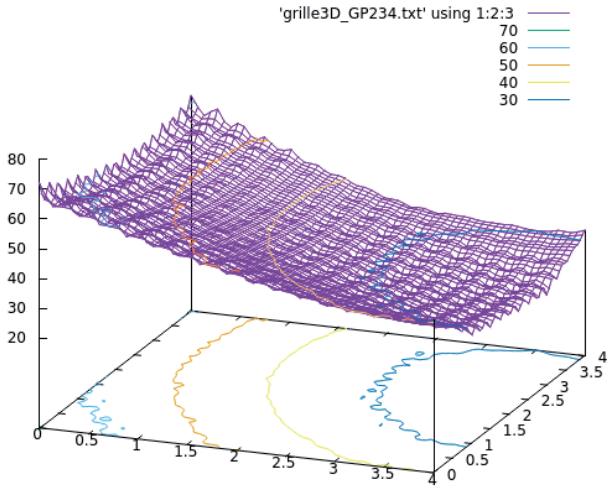
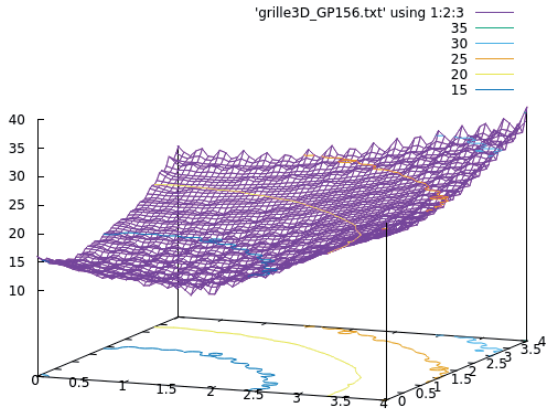
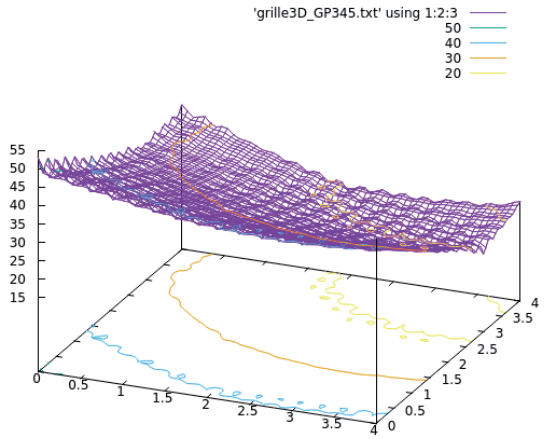


Fig-IV-D3-46/49









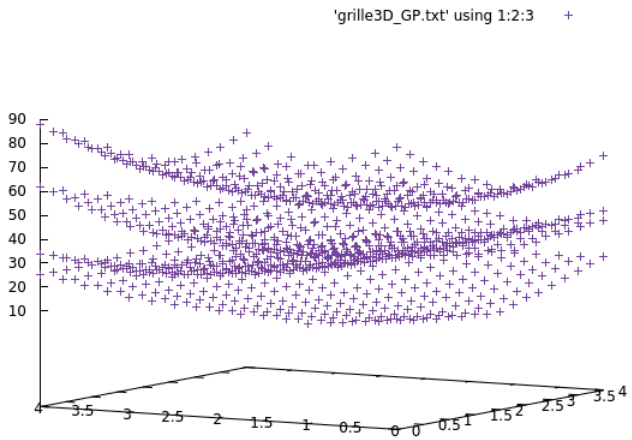


Fig-IV-D3-50/54

```
gnuplot> set contour both
gnuplot> set hidden3d
gnuplot> set dgrid3d 60,60,2
gnuplot> set isosamples 30,30
gnuplot> set surface
gnuplot> splot 'grille3D_GP.txt' using 1:2:3 w l
```

Le dernier outil réalisé pour ces études, sur *oRis*, est un calculateur d'«états utiles» et de coûts globaux optimaux (CG^*) des multi-points considérés pour une grille donnée. Le cœur du programme consiste à parcourir la grille pour trouver tous les multi-points recherchés et calculer pour chacun d'eux la valeur du coût (CG) correspondant.
Il faudrait faire un calcul récursif pour paramétrer la profondeur choisie, qui correspond au nombre de points du multi-points. Nous avons juste fait quatre programmes distincts («calculCG_ETATS1point.ors», «calculCG_ETATS2points.ors», «calculCG_ETATS3points.ors» et «calculCG_ETATS4points.ors», pour traiter les multi-points avec respectivement un, deux, trois et quatre points.
Le principe retenu est toujours le même. On déclare dans le fichier texte («clientEssaiMP.txt») le nombre de clients, leurs coordonnées sur la grille et la valeur de leur demande en ressources.

On définit les dimensions de la grille dans le programme de calcul via la constante *D_GRILLE*. La sortie des résultats se fait dans un ou plusieurs fichiers en fonction du nombre de points des multi-points :

- avec un point, on obtient le fichier «1points.txt», et pour chaque ligne, les coordonnées (x/y) du point sur la grille et la valeur du coût (CG) associé ;
- avec deux points, le fichier «fc1Bipoints.txt» se compose, pour chaque ligne, du N°, des coordonnées (x_1/y_1/x_2/y_2) des points et du coût (CG) du bipoints. Alors que le fichier «fc1Etats.txt» se compose, pour chaque ligne, du numéro de l'état correspondant au bipoints et de son coût global optimal (CG^*) ;
- avec trois et quatre points par multi-points, il y a respectivement, en sortie, uniquement les fichiers des multi-points «3points.ors» et «4points.ors» ; avec, pour chaque ligne le N°, les coordonnées (x_1/y_1/x_2/y_2…) et le coût (CG) du multi-points associé.

Une partie du code de ces outils peut être utilisé dans les simulations, lorsque les heuristiques utilisent, localement, des calculs exacts pour trouver plus rapidement des minima locaux. C'est le cas par exemple des algorithmes de descente de gradient.

IV-D4) DEUX EXEMPLES SIMPLES D'EXPÉRIMENTATIONS : C_6S_2 ET LA LETTRE 'A'

Nous allons repartir d'exemples vus au chapitre III pour leur appliquer les heuristiques basiques et faire une première analyse des résultats en nous appuyant sur les résultats théoriques et les calculs exacts vus dans ce même chapitre III. En première approximation nous allons comparer, sur les exemples simples de type client/serveur C_iS_j, et la lettre 'A' (alphabet latin), les résultats exacts (en mode continu et discret) avec ceux issus des expérimentations appuyées sur les heuristiques.

PROBLÈME DU CLIENT-SERVEUR C_6S_2

Comme introduit ci-avant, nous regardons le problème de la résolution de type Client/Serveur (C_iS_j), où nous avons six clients (i=6) et deux serveurs (j=2). Nous avons pris cet exemple au chapitre III, pour confronter les résultats des heuristiques à ceux du calcul exact.

Nous allons donc nous confronter aux trois approches suivantes :

1. *discret-1* : les serveur se positionnent sur les points de la grille et le calcul du coût global (CG) est fait aussi sur la grille ;
2. *discret-2* : les serveurs se positionnent toujours sur les points de la grille, mais les calculs de CG sont faits dans l'espace euclidien à deux dimensions (réels) ;
3. *continu-1* : les serveurs se positionnent de façon continue dans l'espace euclidien à deux dimensions, suivant les heuristiques choisies. Les calculs de coût global (CG) sont aussi faits dans cet espace euclidien.

Nous travaillons sur une grille de vingt-cinq (5x5) points, allant du point (0/0) au point (4/4). Les coordonnées des clients (deux premiers paramètres), ainsi que leur besoin en ressources (troisième paramètre) sont décrits pour *oRis* dans le fichier «CreerGrille5x5C6_466422.ors», qui est appelé par le fichier «calculGrilleGC.ors» :

```
// création des clients voulus
Client c1 = NEW Client(2,0,4);
Client c2 = NEW Client(4,1,6);
Client c3 = NEW Client(4,3,6);
//Client c4 = NEW Client(2,4,4);
//Client c5 = NEW Client(0,3,2);
Client c6 = NEW Client(0,1,2);
```

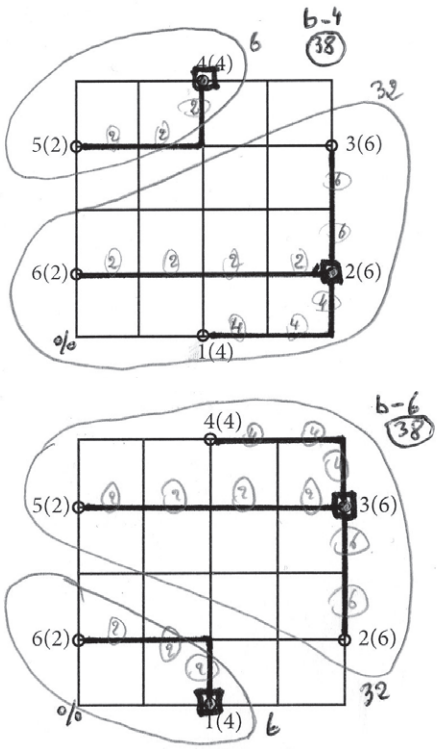


Fig-IV-D4-55/56

On remarque que les clients c4 et c5 sont mis en commentaire (//). Cela correspond au fait que les calculs des coûts globaux (CG) sont faits uniquement avec les clients c1, c2, c3 et c6. Cela correspond à l'une des deux régions de la solution optimale. Pour connaître les coûts pour l'autre région de cette solution optimale, il faudra inverser les commentaires en les déplaçant sur les clients traités dans le cas précédent.

Discussion sur les trois approches sus-mentionnées :

- approche *discret-1* : nous avions vu, au chapitre III (cf. la figure *Fig-III-A5-17*), que les deux configurations optimales (à cause de la symétrie du problème) étaient b-4 et b-6 pour un coût global $CG^*=38$. Sur les figures *Fig-IV-D4-55/56* nous avons les deux serveurs positionnés (carrés noirs) de façon optimale. Ils forment les deux multi-points optimaux $MP^*_{2\text{-}1}$ et $MP^*_{2\text{-}2}$, respectivement pour b-4 et b-6, avec $CG^*_{6,2}=38$, $MP^*_{2\text{-}1}=\{(2,4),(4,1)\}$ et $MP^*_{2\text{-}2}=\{(2,0),(4,3)\}$;

- approche *discret-2*: nous utilisons les programmes «CreerGrille5x5C6_466422.ors» et «calculGrilleCG.ors» pour calculer les valeurs exactes de CG dans l'espace euclidien à deux dimensions. Le deuxième programme appelle le premier qui configure la grille choisie (dimension, ici TAILLE=5, et MAILLE), crée les clients avec leur positionnement et leur besoin en ressources, et lance les calculs des coûts (CG), avant de créer le visuel des résultats. Dans notre exemple les résultats se sont affinés par rapport à l'approche *discret-1*, dans la mesure où la distance euclidienne est potentiellement plus courte qu'en passant par la grille. Nous étions parti de la configuration b-6. Le changement de calcul nous a fait passer dans la configuration c-1 (au chapitre III, cf. la figure *Fig-III-A5-17*), où nous n'avons plus deux clients d'un côté et quatre de l'autre ($Conf_{2,4}$), mais trois de part et d'autre ($Conf_{3,3}$). Comme nous pouvons le voir sur la figure *Fig-IV-D4-57* (cas c-1), le coût global optimal ($CG^*_{6,2}$) vaut 32.62 - il se décompose en 11.62… pour la première région et en 20.94… pour la seconde - contre 38 dans l'approche *discret-1*. Les multi-points optimaux restent identiques. On peut affiner ces résultats en faisant jouer la maille (variable *MAILLE*) de la grille, lors des calculs. Plus cette maille est faible (1,½,¼,⅛,…), plus les calculs seront précis, mais plus ils seront longs. Pour illustrer cela (cf. les figures *Fig-IV-D4-58/65*), nous avons réalisé les calculs pour la configuration b-4 suivant les quatre valeurs 1, ½, ¼ et ⅛ de la maille, sur chacune des deux régions. Ces figures précisent les valeurs des multi-points de la grille en fonction de la précision des positionnements des serveurs;

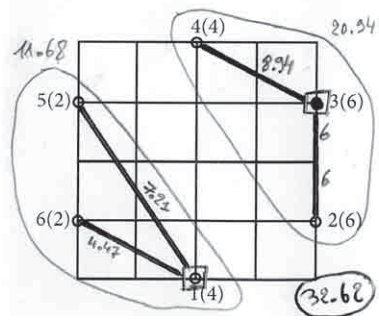


Fig-IV-D4-57

- approche *continu-1*: ce sont les heuristiques expérimentales qui pilotent la recherche des positions optimales en déplaçant les serveurs dans l'espace euclidien à deux dimensions.

Les résultats présentés correspondent donc aux heuristiques proposées dans ce chapitre IV. Comme nous l'avons vu précédemment, les expérimentations peuvent être configurées grâce aux constantes du fichier «fusionV3.cstes». Dans certains cas il s'agit d'utiliser tel ou tel algorithme, lors de la résolution du problème. Nous reviendrons en détails sur ces différentes possibilités dans la suite de ce chapitre IV (partie IV-D).

Voici les principaux éléments utilisés dans la configuration de l'expérimentation (C_6S_2) présentée ci-après:

- les serveurs calculent automatiquement (grâce au positionnement de *rechercheAutoPO* à *true*) et en coopérant entre eux (positionnement de *avecCooperation* à *true*), la valeur de leur période d'observation (PO). Cela signifie qu'elle n'est ni fixe, ni connue au départ de la résolution. Cela signifie également que les serveurs échangent entre eux la valeur de leur propre PO, afin de converger vers ce que nous avons appelé (dans la partie IV-B5) la période d'observation nominale (PoN_k), et ainsi accélérer la synchronisation des serveurs,

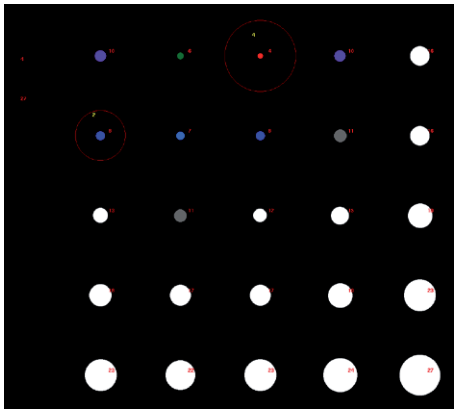


Fig-IV-D4-58

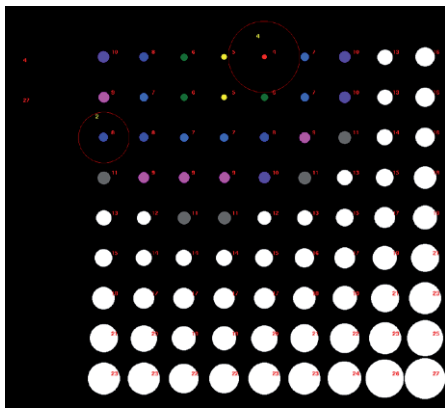


Fig-IV-D4-59

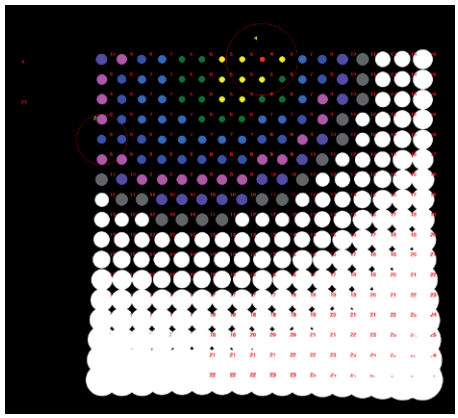


Fig-IV-D4-60

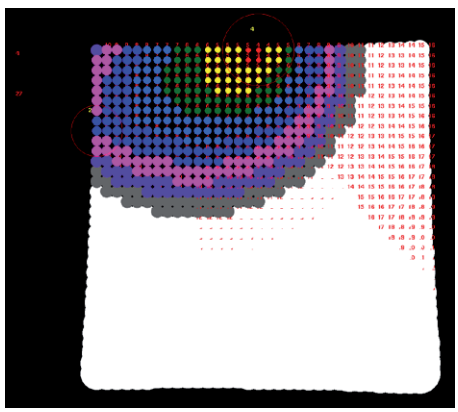


Fig-IV-D4-61

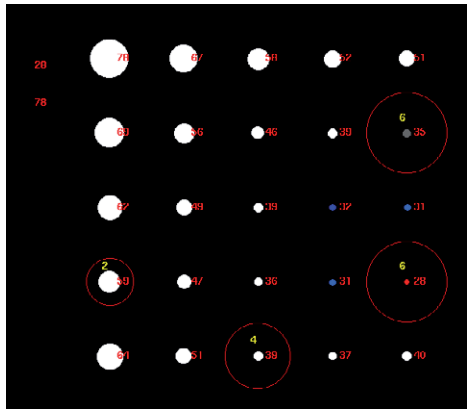


Fig-IV-D4-62

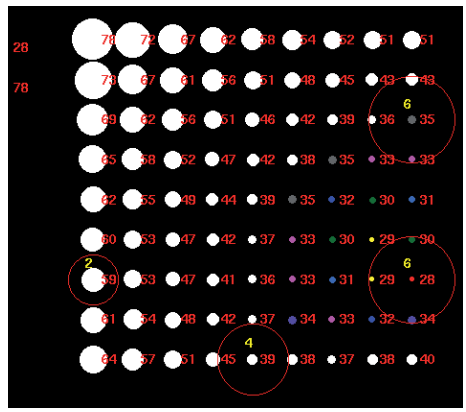


Fig-IV-D4-63

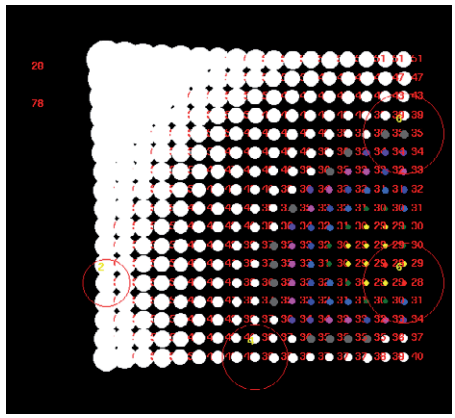


Fig-IV-D4-64

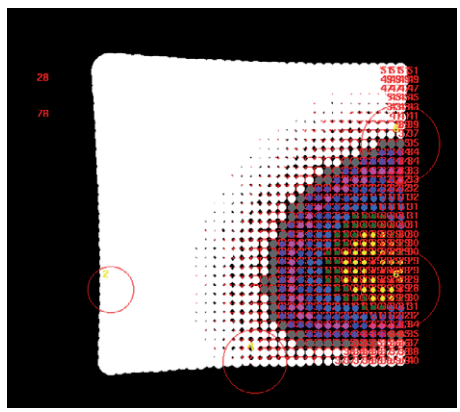


Fig-IV-D4-65

indispensable à la construction d'un état stable du système C_iS_j;
- les serveurs se déplacent (positionnement de *RESTRUCT* à *true*) dans l'algorithme de réorganisation (contraction/expansion), sans utiliser le marcheur aléatoire (positionnement de *MARCHE_ALEA* à *false*), ni la descente de gradient (positionnement de *DESC_GRADIENT* à *false*);
- les requêtes ne peuvent pas repasser (positionnement de *avecRemiseSRVpourReq* à *false*) par un serveur déjà visité et avec lequel elles ont été en échec (incapacité du serveur à répondre à la demande en ressources). Dans le cas contraire les requêtes peuvent revisiter un serveur déjà visité, mais pas deux fois de suite. Cela évite des situations de cycles infructueux pendant lesquels une requête ne serait jamais satisfaite;
- la production de nouvelles ressources par les serveurs se fait en fin de PO (positionnement de *PROD_PO_SRV* à *true*). Dans le cas contraire cette production est faite à chaque top d'horloge. La production en fin de PO correspond en fait à une allocation de ressources, par un nœud du réseau, au serveur en question. L'ajustement consiste donc, pour chaque serveur, à anticiper ces futurs besoins au vu de son passé immédiat. Cette notion d'immédiateté est en fait corrélée à la longueur de la PO, qui est pilotée par le serveur lui-même. Chaque serveur choisit ainsi son horizon temporel, et surtout, le fait évoluer pour se synchroniser avec les autres;
- le calcul de la distance (positionnement de *DISTANCE_PONDEREE* à *true*) d'un serveur à ses clients est pondéré par la demande des clients, en quantité de ressources. Dans le cas contraire, le calcul s'arrête à la distance euclidienne;
- en plus de ces constantes, il existe de nombreux seuils qui sont définis pour le fonctionnement des heuristiques. C'est le cas, par exemple, du seuil de contraction (positionnement de *SEUIL_CONTRACT* à 0.05) qui autorise la contraction des serveurs, dans l'algorithme de réorganisation, uniquement si au moins 5% d'échecs (dans notre exemple) sont constatés à la fin de chaque PO. Cela permet d'éviter des contractions inopportunes. D'autres constantes (comme *SEUIL_STABLE*) travaillent sur la notion de stabilité d'un serveur.

On peut faire plusieurs observations au vu des graphiques obtenus sur la figure *Fig-IV-D4-66*.
La première montre que l'expérimentation a atteint un état stable au bout d'un certain temps (environ 10588 tops d'horloge du moteur *oRis*). Il s'agit d'un attracteur, et non des moindres, puisque le système a trouvé la solution optimale en positionnant les deux serveurs sur le multi-points optimal $MP^*_{2\text{-}2}=\{(2,0),(4,3)\}$, vu dans les approches *discret-1* et *discret-2*. La valeur obtenue pour le coût global optimal ($CG^*_{6,2}$) est affichée sur la fenêtre graphique (fond beige). Elle vaut 32.6275104159…
Sur cet exemple simple les heuristiques fonctionnent bien. Nous poursuivrons, par la suite, notre étude sur des exemples plus riches et plus complexes.
Sans entrer dans le détail, la stabilité du système se voit sur les courbes affichées, mais aussi sur les fenêtres

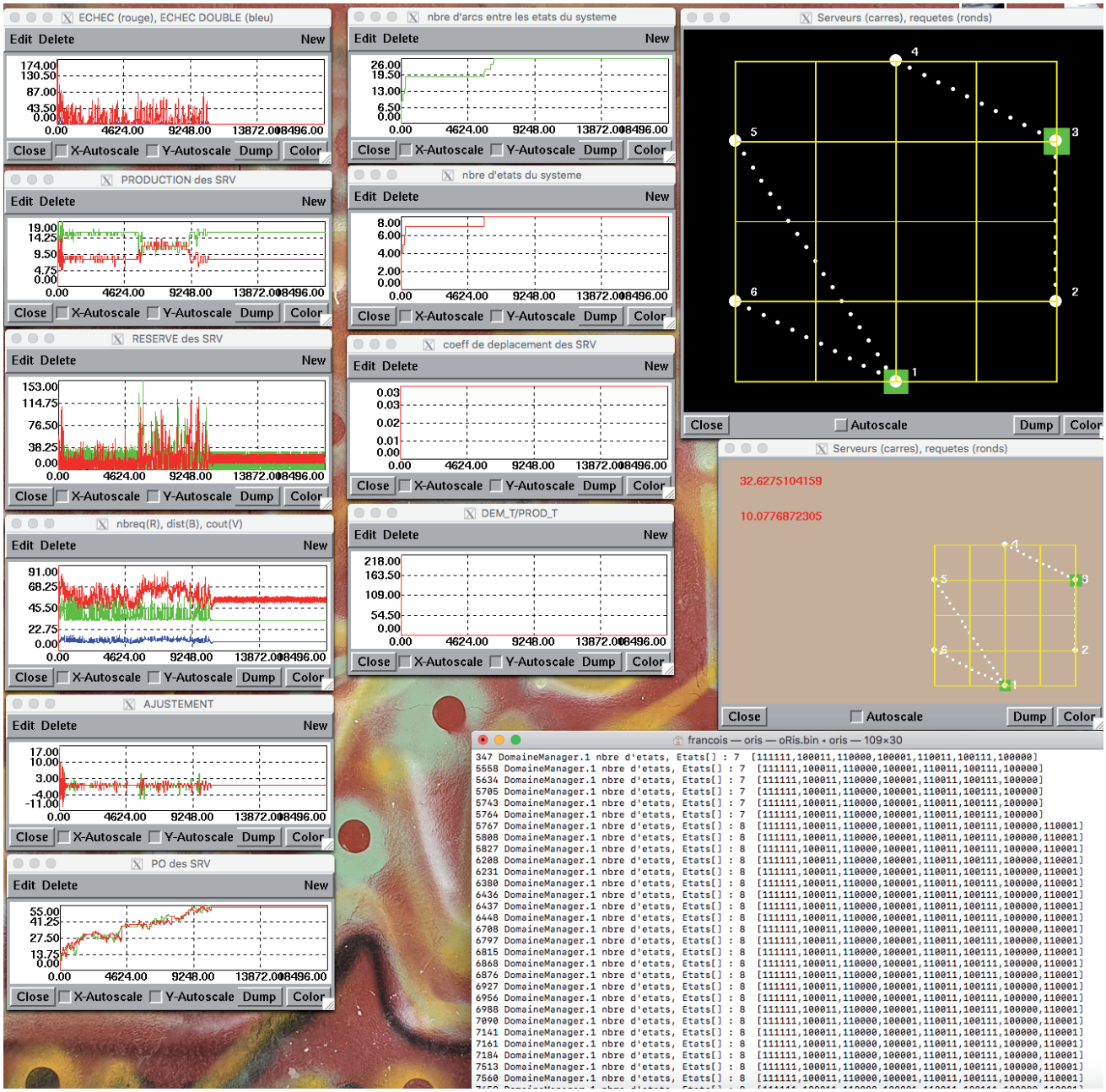


Fig-IV-D4-66

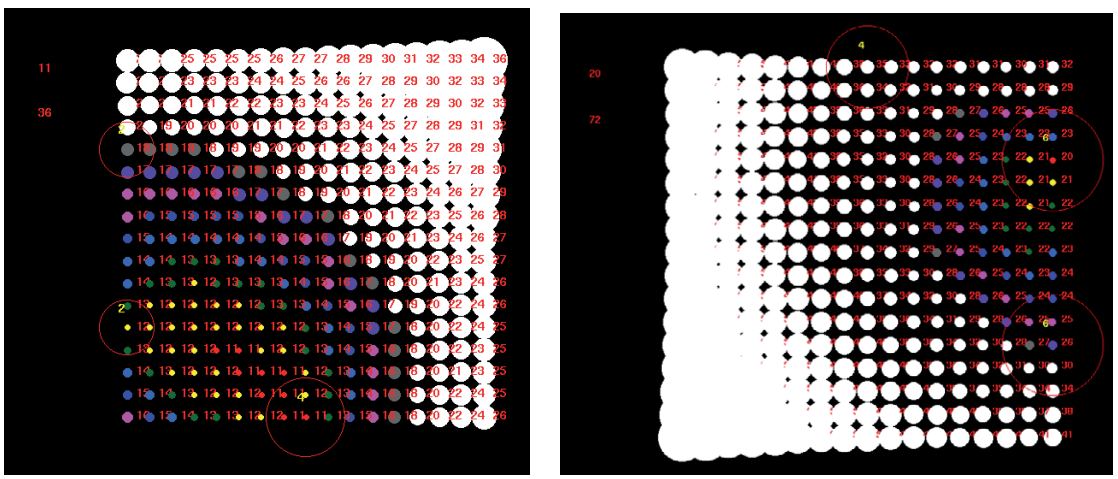


Fig-IV-D4-67/68

graphiques par la matérialisation d'un carré vert, représentant chaque serveur. Ces carrés sont rouges quand les serveurs sont instables, c.-à-d. qu'ils se déplacent et que leur paramètres internes (*production*, *ajustement*, PO,…) évoluent. On voit bien sur les graphiques, qu'à partir du top 10588 (environ), presque toutes les courbes affichent des valeurs constantes.
C'est le cas pour:

- le nombre d'échecs (ECHEC, variable *nbTT_echec* de la classe *Manager*) qui devient égal à zéro. Pour rappel, un échec correspond au fait qu'une requête fait une demande à un serveur qui ne peut la satisfaire. Dans pareil cas, la requête cherche un autre serveur qui pourrait la satisfaire. Cela augmente le nombre de requêtes dans le système. La requête ne disparaîtra du système que lorsqu'elle aura trouvé un serveur qui aura produit la quantité de ressources que son client attend;
- Le nombre de requêtes (*nbreq*, en rouge sur la figure *Fig-IV-D4-69*) n'est pas constant, car les requêtes sont recréées en permanence et disparaissent une fois satisfaites par un serveur. En fonction du nombre de clients et de serveurs à l'équilibre, on a plusieurs cas qui reviennent périodiquement. D'où l'intervalle, lié à ce phénomène, dans lequel le nombre de requêtes évolue de façon cyclique;
- la distance (*dist*, en bleu sur la figure *Fig-IV-D4-69*, variable *dist_aux_srv* de la classe *Manager*) représente la distance cumulée de tous les clients à leur serveur respectif. À l'équilibre cette valeur vaut 10.075;
- le coût (*cout*, en vert sur la figure *Fig-IV-D4-69*, variable *cout_aux_srv* de la classe *Manager*) représente le coût global, c.-à-d. la distance pondérée par les demandes. À l'équilibre cette valeur vaut 32.6275…;
- la production (*PRODUCTION*, variable *production* de la classe *Serveur*) associée à chaque serveur, est respectivement de 16 (en vert sur la figure *Fig-IV-D4-70*) et 8 (en rouge sur la figure *Fig-IV-D4-70*). Ce qui correspond bien aux demandes des clients correspondants. On peut noter, sur la partie centrale du graphique, une période pendant laquelle des échecs ont eu lieu, perturbant les productions des serveurs et le nombre des requêtes associées;
- la réserve (*RESERVE*, variable *reserve* de la classe *Serveur*) représente la quantité globale de ressources réservées par les serveurs et donc disponibles pour les clients, via les requêtes. Cette quantité globale fonctionne à l'équilibre, comme le nombre de requêtes, sur un intervalle (cf. la figure *Fig-IV-D4-71*) dont les valeurs apparaissent de façon cyclique;
- l'ajustement (*AJUSTEMENT*, variable *ajustement* de la classe *Serveur*) permet de répartir sur une période d'observation (PO) courante, l'écart entre les demandes de la période précédente et les ressources réservées pour cette période précédente. En clair, les serveurs ajustent leur besoin avec un décalage d'une PO. À la stabilité du système, tous les serveurs n'ont plus besoin d'ajuster leur production (ajustement=0), comme on peut le voir sur la figure *Fig-IV-D4-72*. On produit exactement ce qui est demandé;

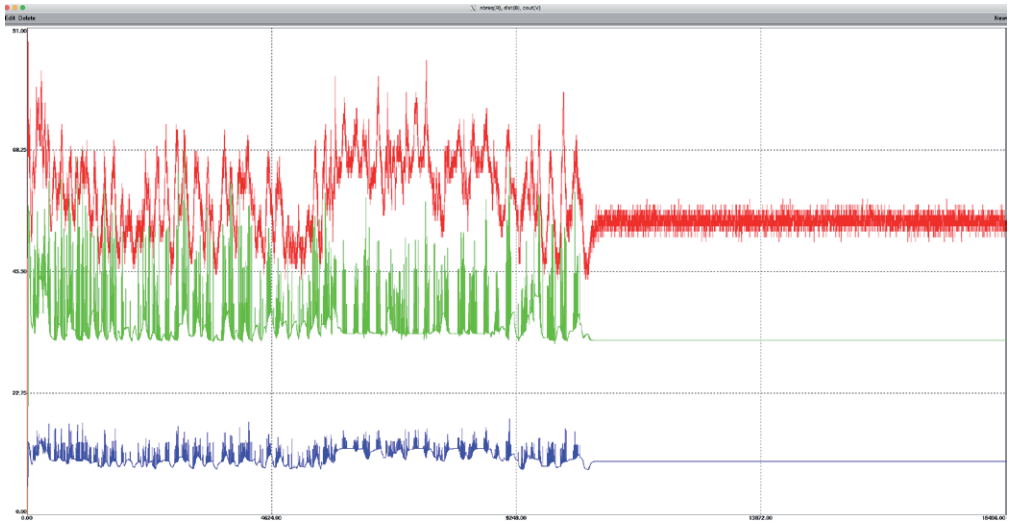


Fig-IV-D4-69

Fig-IV-D4-70

Fig-IV-D4-71

- le rapport entre la demande globale et la production globale (DEM_T/PROD_T, variable *dem_prod* de la classe *Manager*, sur la figure *Fig-IV-D4-74*) est une autre façon de voir ce phénomène d'ajustement, puisqu'il s'agit du rapport entre la demande globale de tous les clients, sur la production globale de tous les serveurs. On peut remarquer, sur la figure *Fig-IV-D4-74*, que cette quantité passe à 1 bien avant la stabilité du système, pour y rester à partir de cet état. En fait, dans ce cas simple, les serveurs subissent de gros ajustements (cf. la figure *Fig-IV-D4-72*) en début de résolution. Cette période correspond bien à *dem_prod* très grand. C'est le moment où les serveurs n'ont pas assez de recul pour s'ajuster à la demande. Très rapidement cela sera fait, comme nous l'avons montré dans la partie IV-B3;
- la période d'observation (PO, variable *periode_obs* de la classe *Serveur*) est un élément décisif dans le processus de résolution, puisque chaque serveur recherche la valeur de sa propre PO afin d'améliorer la convergence du système. Dans l'exemple présenté (C_6S_2) la recherche de la solution intègre la recherche de cette valeur (PO) par une coopération entre les serveurs. On le voit très bien sur la figure *Fig-IV-D4-73*, où les courbes verte et rouge s'entrecroisent suivant une courbe globale croissante, qui marque la coopération entre les serveurs (échange de PO). À l'équilibre, ces deux courbes n'en font plus qu'une, constante, correspondant à la période d'observation nominale (PoN_k), ici égale à 54, pour les deux serveurs en jeu. On voit aussi, par moments, que chaque serveur peut s'écarter de la PO de l'autre, pour y revenir ultérieurement;
- le nombre d'états du système (variable *Etats* de la classe *DomaineManager*) correspond aux états dans lequel le système est passé pendant la résolution. Nous rappelons qu'un état correspond à une répartition des clients sur les serveurs. Là encore on constate qu'à l'équilibre, le système ne traverse plus de nouveaux états, puisqu'il est stable dans l'état où la résolution est optimale;
- le nombre d'arcs entre les états du système (variable *nb_arcs_dom* de la classe *DomaineManager*). Le nombre d'arcs, entre états, est lui aussi constant à l'équilibre. Un arc, entre deux états, signifie que le système est passé d'un état à l'autre lors de la résolution. Là encore cette valeur devient constante dès que le système est stable, puisqu'il arrête d'explorer l'espace des états.

Représentation de l'alphabet : la lettre 'A'

Comme nous l'avons introduit dans la partie III-B2, les lettres de l'alphabet forment un deuxième exemple intéressant pour appliquer nos heuristiques.
Comme nous le verrons par la suite, l'alphabet va nous permettre d'appliquer la résolution C_iS_j, non plus comme un simple calcul d'optimalité, mais comme une technique de représentation des connaissances/encodage fondée sur une optimisation de l'utilisation des ressources mobilisées à cet égard.

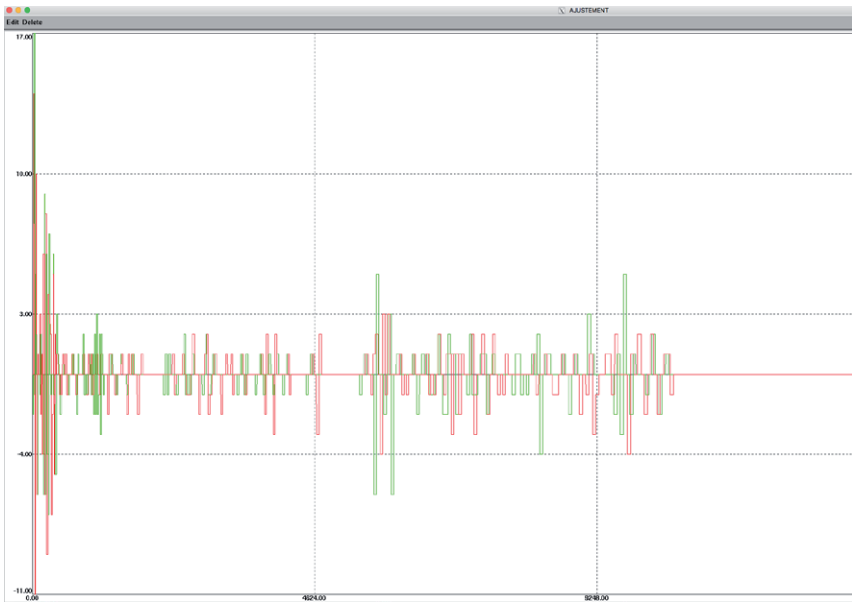


Fig-IV-D4-72

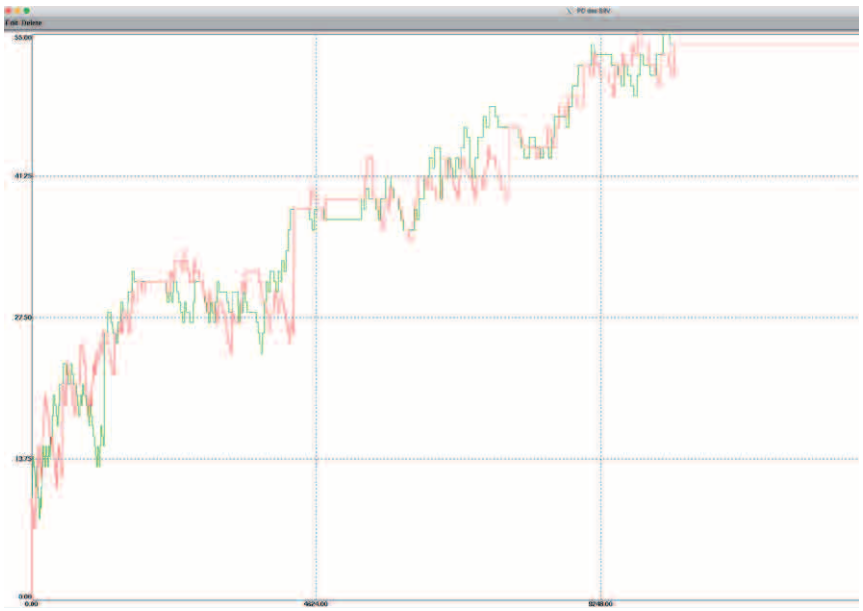


Fig-IV-D4-73

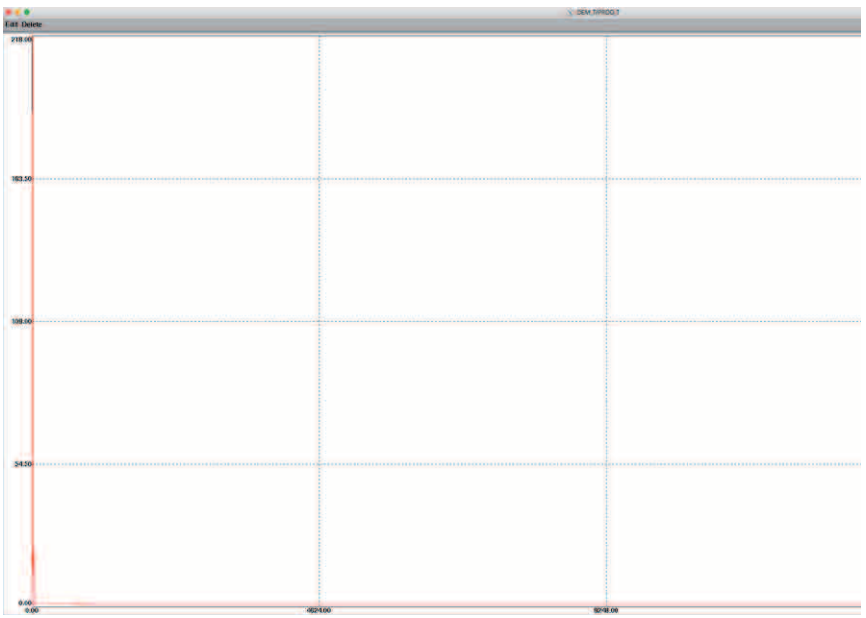


Fig-IV-D4-74

Nous allons regarder, dans un premier temps, comment les heuristiques résolutives se comportent avec la lettre majuscule 'A' de l'alphabet latin.

Pour comparer les résultats avec l'étude théorique, nous prenons la lettre 'A' composée de vingt-cinq points-motif (clients) sur une grille de trente-six (6x6) points (D=6), conformément aux figures *Fig-IV-D4-75/76*.
Nous utilisons deux serveurs (carrés violets superposés sur la figure *Fig-IV-D4-76*), positionnés, initialement, au centre de la lettre. Les calculs exacts de la partie III-B2 sont réalisés sur une grille, avec une maille plus ou moins fine. Les expérimentations fonctionnent en continu, dans un espace euclidien en deux dimensions.

```
manager_domaine->creer_Client("1",0, 0,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",0.25, 0.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",0.5, 1,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",0.75, 1.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",1, 2,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",1.25, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",1.5, 3,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",1.75, 3.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2, 4,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2.25, 4.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2.5, 5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2.75, 4.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",3, 4,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",3.25, 3.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",3.5, 3,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",3.75, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",4, 2,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",4.25, 1.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",4.5, 1,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",4.75, 0.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",5, 0,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",1.75, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2.25, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",2.75, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);
manager_domaine->creer_Client("1",3.25, 2.5,"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50, manager);

manager_domaine->creer_Serveur("1", 2.5, 2.5,"m","dr",1,10, manager, PREMIER);
manager_domaine->creer_Serveur("1", 2.5, 2.5,"m","dr",1,200, manager, PREMIER);
```

Fig-IV-D4-75

Nous avons réalisé une première expérimentation dans la même configuration qu'avec C_6S_2, vu précédemment. Les calculs exacts nous donnaient un coût optimal ($CG^*_{25,2}$) de valeur 33.69… avec le bipoints optimal $MP^*_{2\_1}=\{(1,2),(3,3)\}$. Dans cette première expérimentation (cf. les figures *Fig-IV-D4-77/82*) nous obtenons un système stable (à l'équilibre) au top 12632 (environ) avec un coût optimal ($CG^*_{25,2}$) de valeur 33.7367… et le bipoints optimal correspondant $MP^*_{2\_1}=\{(1,2),(3.5,3)\}$.

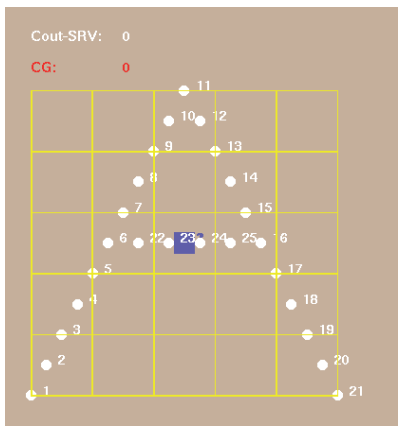


Fig-IV-D4-76

La deuxième expérimentation reprend la première en ajoutant la descente de gradient (la constante *DESC_GRADIENT* vaut *true* dans le fichier «fusionV3.cstes») des serveurs, lors de la recherche de l'optimum local. Il s'agit, pour les serveurs, d'intégrer les gradients de coûts CG lors de leurs déplacements à la recherche de l'optimum local. On a montré, au chapitre III, qu'une partition s'organisait avec un ensemble de clients et d'un serveur avec un ou plusieurs optima locaux (là où le coût CG est le plus faible) et un gradient conduisant à ces optima.

Comme on peut le voir sur notre exemple (cf. les figures *Fig-IV-D4-83/88*), cette propriété additionnelle fait converger plus rapidement la résolution (état stable du système). En effet au top 6077 (environ) le système est stable avec un coût optimal (CG^*) qui vaut 33.4845…, meilleur que le précédent, avec comme bipoints optimal correspondant $MP^*_{2\_1}$={(1,2),(3.25,3)}.

Comme nous l'avons évoqué précédemment, il est logique que le coût global optimal soit meilleur qu'avec le calcul exact, qui travaille, dans notre exemple, sur une grille avec une maille de ½. Ici les serveurs se déplacent en continu, indépendamment de la grille sous-jacente.

On peut faire quelques commentaires sur les deux expérimentations :

- on voit par exemple que les productions des serveurs (cf. les figures *Fig-IV-D4-79* et *Fig-IV-D4-85*) sont différentes dans les deux cas : 17 pour l'un, 8 pour l'autre avec gradient et 15 pour l'un et 10 pour l'autre sans cette descente. Cela se voit sur les figures *Fig-IV-D4-77* et *Fig-IV-D4-83* où les points-motif n°8 et n°23 sont passés du serveur n°1 au serveur n°2, provoquant un changement d'état. Lors de la résolution sans descente de gradient, le système est passé par l'état où il avait atteint sa stabilité avec la descente de gradient ;
- si l'on regarde le nombre de requêtes (cf. les figures *Fig-IV-D4-80* et *Fig-IV-D4-86*) on voit que sans la descente de gradient, le système avait presque atteint une stabilité en début de courbe (entre les tops 2000 et 4000 environ), avant de rentrer dans une zone d'instabilité. Toutes les courbes montrent le phénomène suivant où l'on voit que l'expérimentation en cours, sans descente de gradient, vient de traverser, sans s'y arrêter, l'état final de l'expérimentation avec descente de gradient. Le système a changé d'état sans aller explorer complètement l'état qu'il vient de quitter. La descente de gradient a corrigé ce problème. Dans les cas plus complexes, ce mécanisme peut se retourner contre la recherche de l'optimum en tombant dans un attracteur local. D'autres mécanismes viendront corriger alors cette difficulté ;
- les périodes d'observation (PO) sont calculées, dans les deux cas, automatiquement et par coopération entre les serveurs. Sans descente de gradient les deux serveurs partagent la même valeur (13) de PO dans la première période de pseudo-stabilité (entre les tops 2000 et 4000). Ensuite la coopération fait évoluer ces valeurs de façon distincte pour chaque serveur, mais en suivant une courbe globale commune. À la vraie stabilité du système, les deux serveurs retrouvent une valeur commune (48). Avec la descente de gradient le résultat de la coopération est beaucoup plus linéaire. À la résolution, les deux serveurs ont des périodes d'observations différentes, mais voisines (53/55).

Là encore, on voit que les heuristiques résolutives fonctionnent bien dans des cas simples. En sera-t-il toujours ainsi avec des cas plus complexes (viabilité au passage à l'échelle) ?

Sans descente de gradient

top 12632 (environ)

$CG^*_{25,2}$=33.7367…

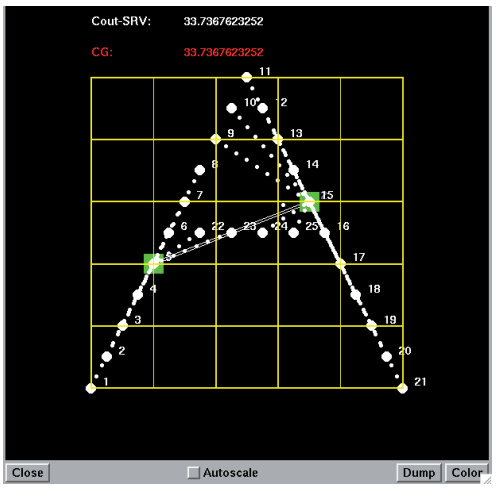
$MP^*_{2\_1}$={(1,2),(3.5,3)}

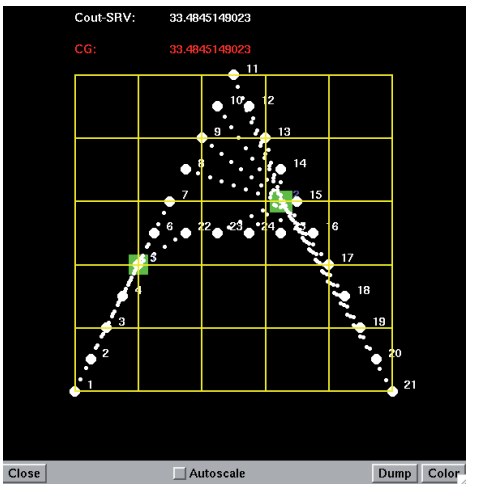
Avec descente de gradient

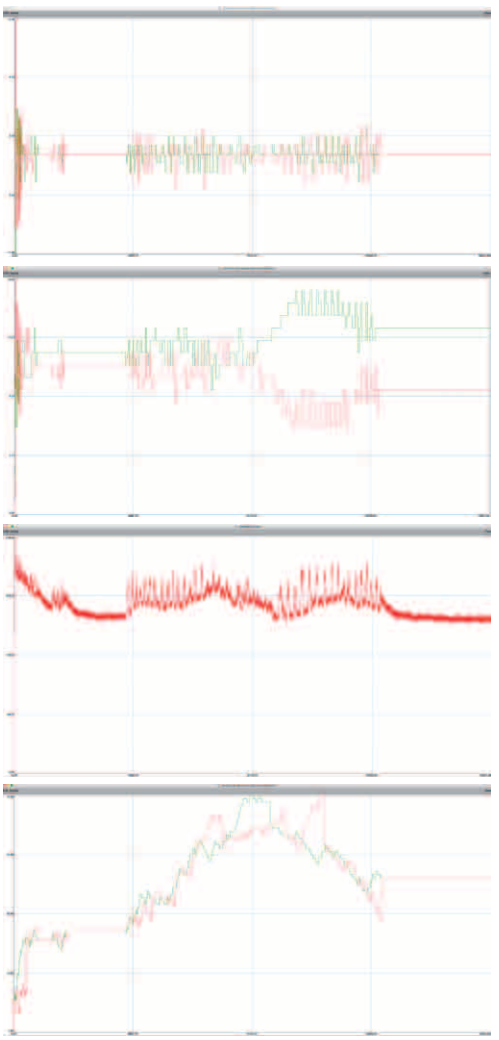
top 6077 (environ)

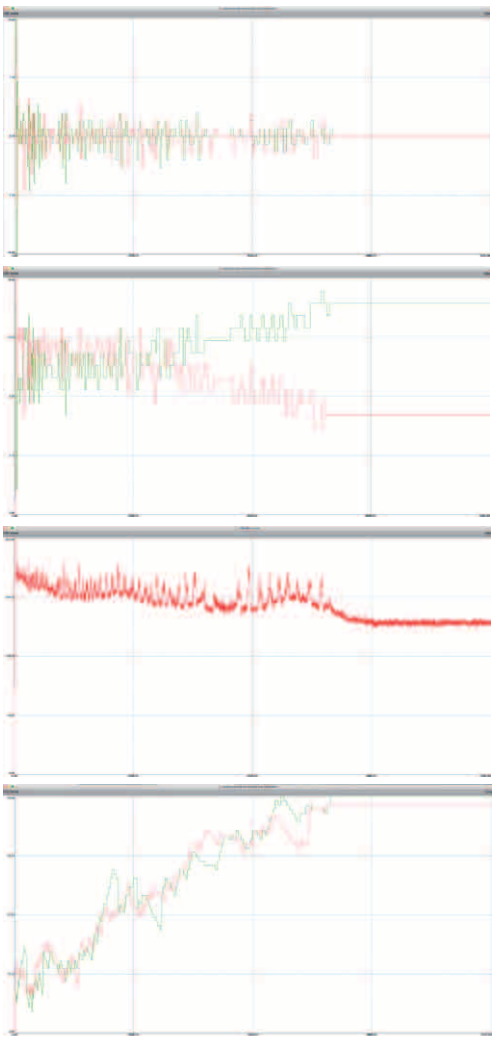
$CG^*_{25,2}$=33.4845…

$MP^*_{2\_1}$={(1,2),(3.25,3)}









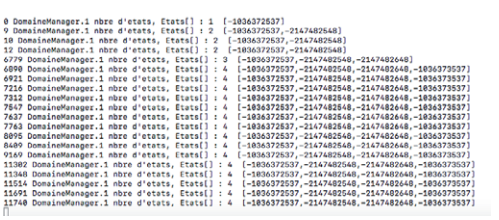


Fig-IV-D4-77/82

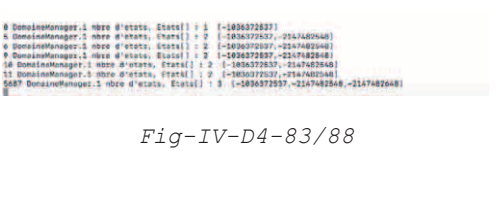


Fig-IV-D4-83/88

Les algorithmes utilisés dans les heuristiques résolutives peuvent être configurés/paramétrés par les constantes proposées dans le fichier «fusionV3.cstes».

Des adaptations du code peuvent aussi être réalisées conformément aux résultats théoriques décrits au chapitre III.

C'est ce que nous allons voir dans les parties suivantes.

Simulations : Expérimentations et résultats

IV-E

Dans le chapitre VI nous aurions du aborder le problème de l'instanciation et de la différenciation de modèles. Pour cela nous aurions travaillé sur l'alphabet latin en lettres majuscules, dans la suite logique des travaux déjà abordés avec la lettre 'A'.
Pour cela nous aurions aussi utilisé les résultats sur la résolution C_iS_j pour l'engrammage des modèles. Nous y aurions ajouté les résultats sur la dérive des connaissances, que nous aurions présentés au chapitre V, plus d'autres développements.

La résolution efficiente C_iS_j aurait pu apparaître comme une piste prometteuse pour engrammer de façon optimale (en terme de ressources utilisées) des connaissances, au service de mécanismes de plus haut niveau pour la représentation de connaissances incluant les modèles.

Cette partie IV-E nous donne l'occasion d'approfondir les heuristiques résolutoires en pointant des problèmes soulevés dans les résultats théoriques, au chapitre III, mais aussi dans les premières version de notre simulation (jusqu'à la v3).

Nous allons commencer par regarder ce qu'il advient de la pertinence des heuristiques (convergence vers un optimal) dans des cas plus «riches» que ceux vus précédemment. En particulier nous mettons l'accent sur des cas qui nous seront utiles pour résoudre les problèmes posés au chapitre VI (représentation des connaissances).

Cette partie va aussi être l'occasion de revoir les mécanismes profonds d'affichage pour le suivi en temps réel des simulations. Cela va nous conduire à proposer la version v5 de la simulation.

Nous allons aussi aborder la question de l'exploration de l'espace des états, dans des situations plus complexes, en proposant une version complètement nouvelle (version v6), car enrichie, du moteur comportemental des serveurs. Cette version permet une exploration plus riche et surtout plus pertinente de cet espace des états, en ajoutant une dimension autopoïétique au dispositif.

Nous concluons cette partie sur des résultats très intéressants pour la suite de nos travaux.

IV-E1) $C_I S_J^{++}$…

Nous allons utiliser des expérimentations avec davantage de clients que précédemment, et regarder comment réagissent nos heuristiques avec un nombre variable de serveurs pour une grille donnée. Nous confronterons les résultats des calculs exacts avec nos résultats expérimentaux.

Vingt-et-un clients et deux serveurs ($C_{21}S_2$)

Dans ce premier exemple nous partons avec vingt-et-un clients et deux serveurs sur une grille de deux-cent-vingt-cinq (15x15) points (D_GRILLE=15). Nous lançons, dans *oRis*, le fichier «calculCG*ETATS2points.ors», qui calcule le coût global optimal (CG^*) en nous appuyant sur le fichier «clientEssai.txt» (cf. la figure *Fig-IV-E1-89*) pour deux serveurs (bipoints).

Nous obtenons les résultats exacts suivants :
$CG^*_{21,2}$=409.56…,
$MP^*_{2\_1}$={(1,5),(8,13)},
cent-soixante états différents.

En lançant les heuristique sur le même problème nous obtenons les résultats suivants :
$CG^*_{21,2}$=408.50…,
$MP^*_{2\_1}$={(1.25,4.75),(8.25,13.25)},
seulement six états ont été parcourus.

```
21
7 0 2
8 0 5
14 6 4
14 7 6
14 8 2
14 11 2
14 12 4
11 14 3
8 14 5
7 14 7
6 14 3
3 14 4
2 14 3
0 11 2
0 8 6
0 7 8
0 6 5
4 0 7
0 2 6
3 0 2
6 0 2
```

Fig-IV-E1-89

On voit, sur la figure *Fig-IV-E1-90*, que le système a trouvé la solution optimale (cf. la valeur exacte sur la grille), mais n'arrive pas à se stabiliser. Cela est dû aux échecs induisant des interactions résiduelles entre les deux serveurs et une évolution incessante de leur période d'observation (PO).

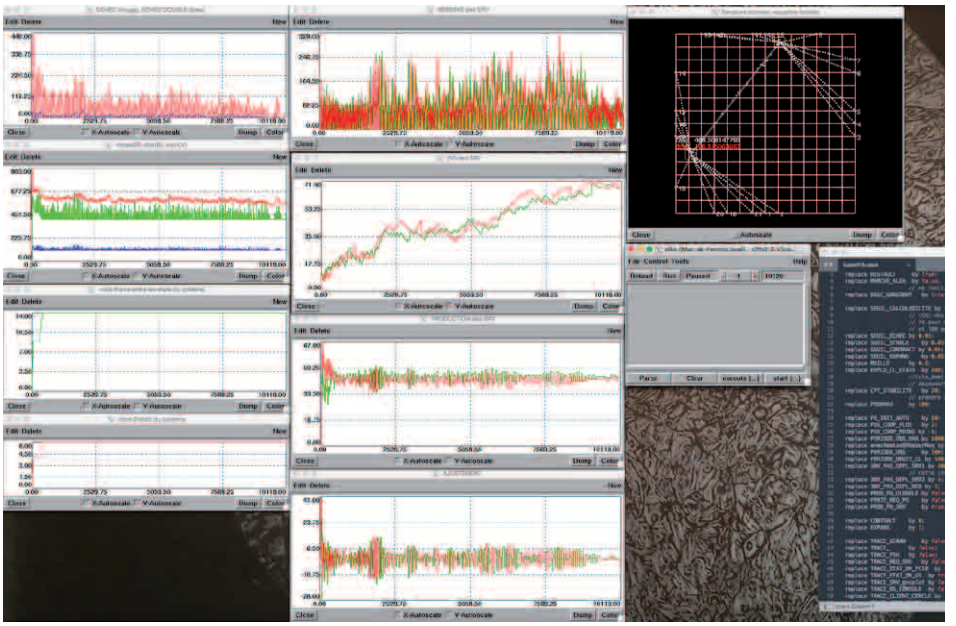


Fig-IV-E1-90

Cet exemple pointe le fait que la recherche d'un état stable n'est pas toujours souhaitable.

Nous verrons, dans la suite des résultats, que la stabilité est intéressante dans certains cas particuliers.

C'est le cas lorsque le système de résolution arrive dans un minimum local. Sa propre perception de cet état de stabilité, lui permet de lancer des algorithmes adhoc, comme par exemple celui de la mitose, appelé encore «division cellulaire».

VINGT CLIENTS ET UN NOMBRE VARIABLE DE SERVEURS ($C_{20}S_j$)

Dans ce deuxième exemple nous partons avec vingt clients et faisons varier le nombre de serveurs pour voir comment se comportent nos heuristiques résolutives.

Cette fois nous utilisons, sur cet exemple, les calculs exacts faits en langage 'C' (au chapitre III). Nous avons calculé les résultats pour les douze premières configurations, c.-à-d. allant d'un serveur à douze.

Nous obtenons les résultats suivants :

j	$CG^*_{20,j}$	MP^*_{j-i}
1	38,7624…	{(2,3)}
2	28,7108…	{(1,3),(4,3)},{(2,1),(2,4)}
3	21,1677…	{(1,2),(3,4),(4,1)}
4	14,5723…	{(1,1),(1,4),(4,1),(4,4)}
5	13,3074…	{(1,1),(1,4),(3,2),(4,1),(4,4)}
6	12,0957…	{(1,1),(1,3),(1,4),(3,2),(4,1),(4,4)} {(1,1),(1,4),(2,4),(3,2),(4,1),(4,4)}
7	10,8839…	{(1,1),(1,3),(1,4),(2,4),(3,2),(4,1),(4,4)}
8	10,0538…	{(1,1),(1,3),(1,4),(2,1),(2,4),(4,1),(4,3),(4,4)}
9	09,5120…	{(1,1),(1,2),(1,3),(1,4),(2,1),(2,4),(4,1),(4,3),(4,4)} {(1,1),(1,3),(1,4),(2,1),(2,4),(3,4),(4,1),(4,3),(4,4)}
10	08,9703…	{(1,1),(1,2),(1,3),(1,4),(2,1),(2,4),(3,4),(4,1),(4,3),(4,4)}
11	08,4463…	{(1,1),(1,2),(1,3),(1,4),(2,1),(2,4),(3,2),(3,4),(4,1),(4,3),(4,4)}
12	08,0758…	{(1,1),(1,2),(1,3),(1,4),(2,1),(2,4),(3,1),(3,4),(4,1),(4,2),(4,3),(4,4)}

Pour éviter le déclenchement de la duplication de serveurs (algorithme de mitose), nous avons passé la constante *PRODMAX* à 100. Cela signifie que la production (réservation de ressources locales) associée à chaque serveur doit dépasser cette valeur pour que ce dernier se clone. Dans notre exemple cette valeur ne sera pas atteinte.

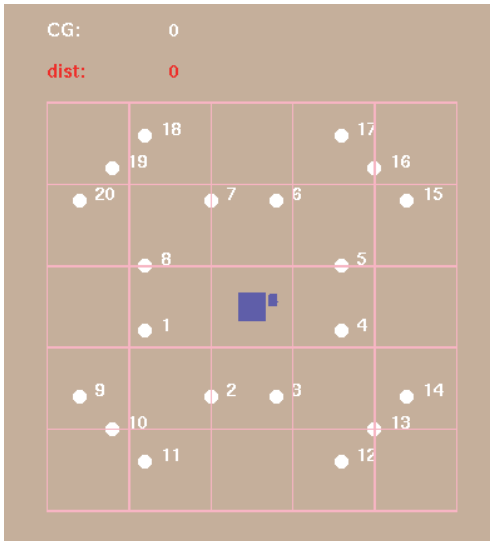


Fig-IV-E1-91

Pour reprendre la précision du calcul exact nous travaillons sur une grille avec trente-six (6x6) points (D_GRILLE=7) et une maille (*MAILLE*) de ½.
Les clients sont donc positionnés conformément à la figure *Fig-IV-E1-91*. Les serveurs sont placés au centre de la grille pour ne pas introduire un biais dû à leur position initiale. En réalité toute position initiale des serveurs introduit nécessairement un biais dans la résolution, à cause de celle des clients. Ce biais peut être effectif dans les résultats obtenus, pour des situations particulières. Ce serait le cas si tous les serveurs partaient du coin bas gauche (0/0). En choisissant le centre de la grille, le biais est, a priori, atténué.

Nous verrons plus loin que l'algorithme de mitose contourne cette notion de biais, dû à la position initiale des serveurs, en le transformant en un «biais utile».
Quand un serveur se clone, il crée un nouveau serveur, là où les besoins (demandes des clients) le requièrent.

Nous avons fait tourner les heuristiques en partant avec un seul serveur, pour aller (via des clonages successifs) jusqu'à huit serveurs.

Nous obtenons les résultats suivants, qui confirment l'efficacité des heuristiques, tout en soulevant une problèmatique propre aux systèmes dynamiques, celle des attracteurs locaux :

- un serveur (cf. la figure *Fig-IV-E1-92*) :
  $CG^*_{20,1}$=37.8345…,
  $MP^*_{1\_1}$={(3.3,2.6)} ;
- deux serveurs (cf. la figure *Fig-IV-E1-93*) :
  $CG^*_{20,2}$=27.7477…,
  $MP^*_{2\_1}$={(3.66,2.49),(1.11,2.82)},
  $CG^*_{20,2}$=27.7236…,
  $MP^*_{2\_1}$={(3.63,2.74),(1.12,2.61)},
  $CG^*_{20,2}$=27.7148…,
  $MP^*_{2\_1}$={(3.62,2.65),(1.14,2.64)}.
  Le système a atteint l'optimal, tout en étant stabilisé (les carrés verts représentent les serveurs stables) ;
- trois serveurs (cf. la figure *Fig-IV-E1-94*) :
  $CG^*_{20,3}$=20.9343…,
  $MP^*_{3\_1}$={(3.74,3.44),(0.99,4.01),(2,1.36)}.
  Le système a trouvé l'optimal, tout en étant stabilisé ;
- quatre serveurs (cf. la figure *Fig-IV-E1-95*) :
  $CG^*_{20,4}$=14.3268….
  Le système a trouvé l'optimal, tout en étant stabilisé ;
- cinq serveurs (cf. la figure *Fig-IV-E1-96*) :
  $CG^*_{20,5}$=12.6711….
  Le système a trouvé l'optimal, tout en étant stabilisé ;

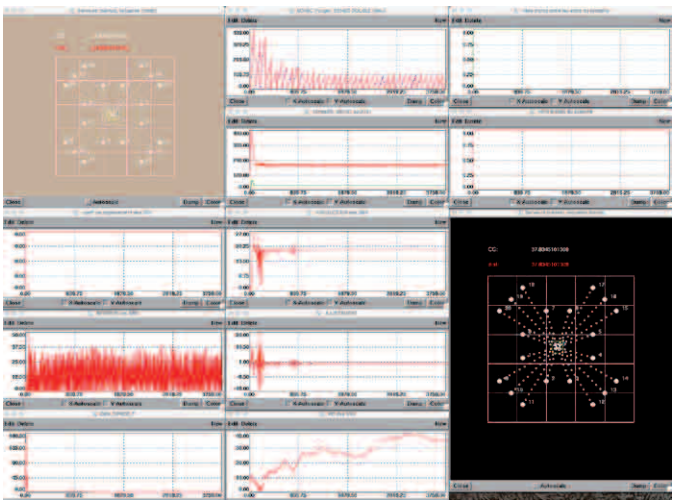


Fig-IV-E1-92

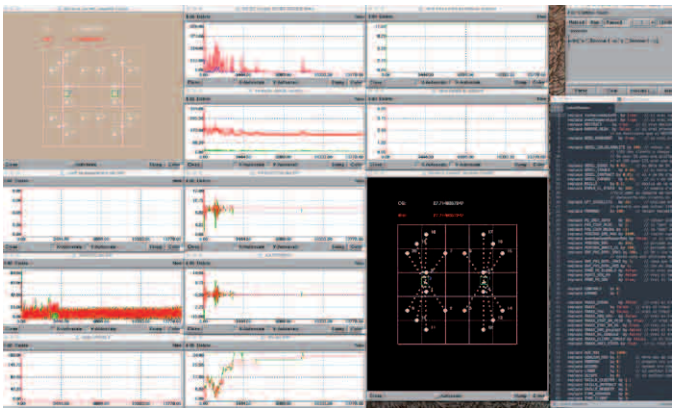


Fig-IV-E1-93

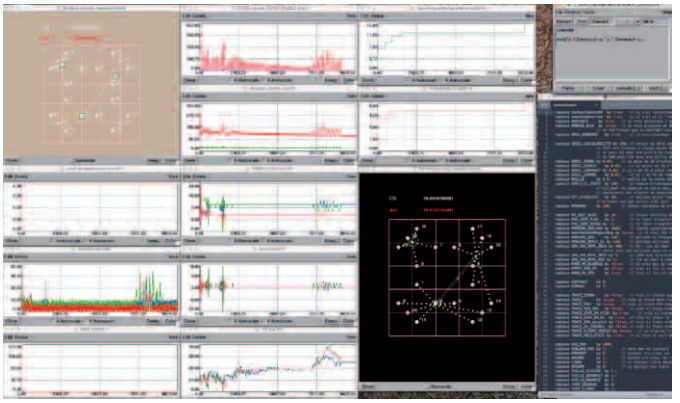


Fig-IV-E1-94

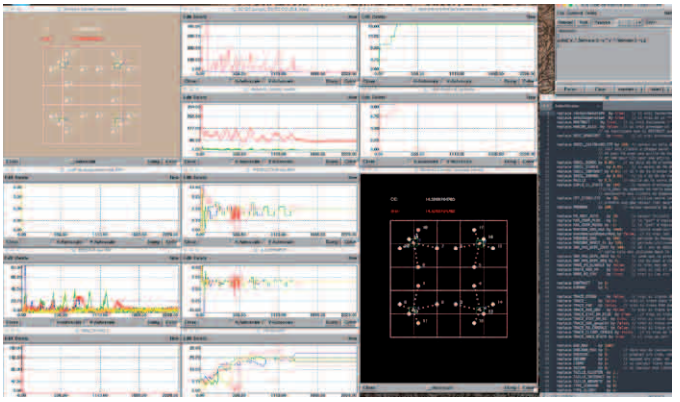


Fig-IV-E1-95

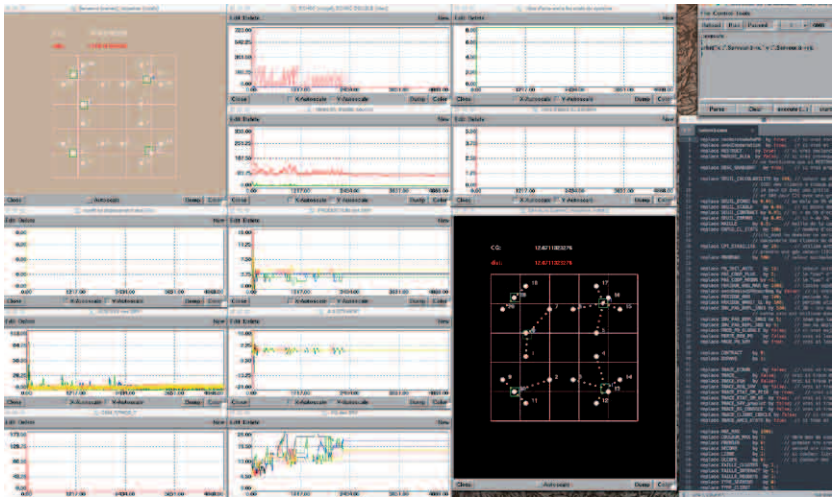


Fig-IV-E1-96

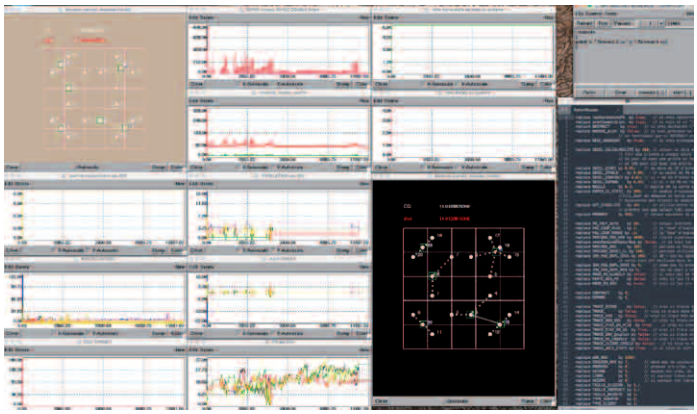


Fig-IV-E1-97

Fig-IV-E1-98

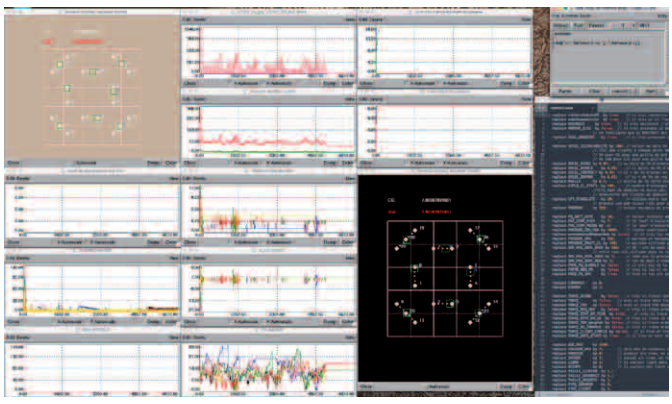


Fig-IV-E1-99

- six serveurs (cf. la figure *Fig-IV-E1-97*) :
$CG^*_{20,6}$=11.0129….
Le système a trouvé l'optimal, tout en étant stabilisé;
- sept serveurs (cf. la figure *Fig-IV-E1-98*) :
$CG^*_{20,7}$=09.4608….
Le système a trouvé l'optimal, mais n'arrive pas à se stabiliser (les carrés jaunes représentent les serveurs instables et leur période d'observation n'est pas constante) ;
- huit serveurs (cf. la figure *Fig-IV-E1-99*) :
$CG^*_{20,8}$=07.9038….
Le système a bien trouvé l'optimal, tout en étant stabilisé.

Là encore on peut constater que les heuristiques résolutives se comportent correctement puisque les optimaux sont atteints dans les huit configurations. Seule la configuration avec sept serveurs n'arrive pas à se stabiliser.
D'autres mécanismes et contraintes (divers seuils, comme par exemple lors des algorithmes de restructuration) peuvent être utilisés pour rechercher l'équilibre (stabilité du système), si celui-ci existe.
Il faut noter que les écarts entre les valeurs exactes et les valeurs obtenues par les heuristiques sont dûs aux calculs continus et non discrétisés sur la grille. En effet les serveurs se déplacent dans l'espace euclidien en deux dimensions et non uniquement sur la grille.

Deux-cents clients et huit serveurs ($C_{200}S_8$)

Pour valider et approfondir les heuristiques sur des exemples plus grands, nous allons utiliser un fichier générateur de problèmes C_iS_j, où 'i' et 'j' sont paramétrables. Nous choisissons deux-cents clients (i=200) et huit serveurs (j=8) sur une grille de deux-cent-vingt-cinq (15x15) points (D=15) et une maille de ½ (MAILLE=½).
C'est une première étape vers le passage à l'échelle, qui va nous conduire à proposer des modifications dans le code de la simulation (passage de la version v3 aux versions v4, puis v5).
Le générateur d'exemple permet de choisir le nombre de clients et de serveurs, avec une position aléatoire. Cela permet de créer des exemples aussi divers que souhaitables, avec l'inconvéniant de ne jamais sortir, du générateur, deux fois le même exemple (à cause des positions tirées aléatoirement) :

```
while(numberOf("Client")<200) manager_domaine->creer_Client
                  ("1",(14*?),(14*?),"nm","dr",1,0.5,50,0.5,-1, 1,-0.5,50,-0.5,-1, 50,manager);
while(numberOf("Serveur")<8) manager_domaine->creer_Serveur
                  ("1",(14*?),(14*?),"nm","dr",1,18,manager,PREMIER);
```

Pour remédier à cet inconvénient, il est toutefois possible de sauvegarder le problème généré (le nombre et les positions des clients et des serveurs) afin de pouvoir le rejouer en utilisant des configurations différentes dans nos heuristiques. Nous modifierons le code pour cela.

Sur la figure *Fig-IV-E1-100*, on voit les huit régions qui se sont constituées autour des huit serveurs. On comprend que l'augmentation du nombre de clients et/ou de serveurs conduirait à un résultat analogue (bonne répartition des clients en régions sur les serveurs).
Les figures *Fig-IV-E1-101/102* montrent, respectivement, la simulation après environ mille, puis quatre-mille tops d'horloge. La valeur du CG calculé (le «coût» représenté par la courbe verte sur les diagrammes) passe de 396 à 378, signifiant que la solution s'améliore. On peut remarquer que le nombre d'états parcourus est de quatre dans les deux cas. Le système trouve relativement rapidement un état proche de l'équilibre avec les huit régions autour des huit serveurs, et n'en sort plus. Soit le système est tombé directement sur la solution optimale, ce qui est peu probable, soit il rencontre une solution et y reste (attracteur local), soit encore il parcourt beaucoup d'états, avec des oscillations possibles entre états proches. Le nombre d'états parcourus, ici quatre, est un indicateur de la dynamique du système de résolution. Peu d'états signifie que nous sommes probablement dans un

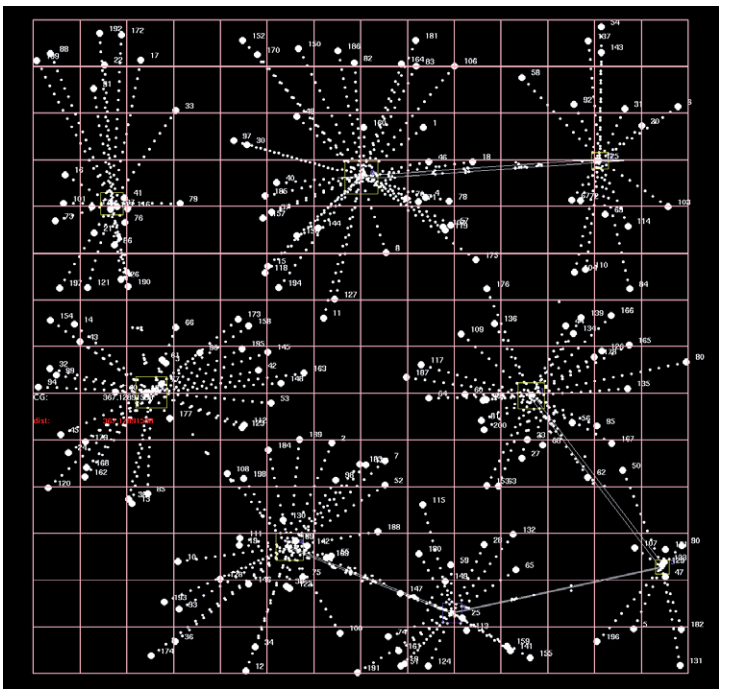


Fig-IV-E1-100

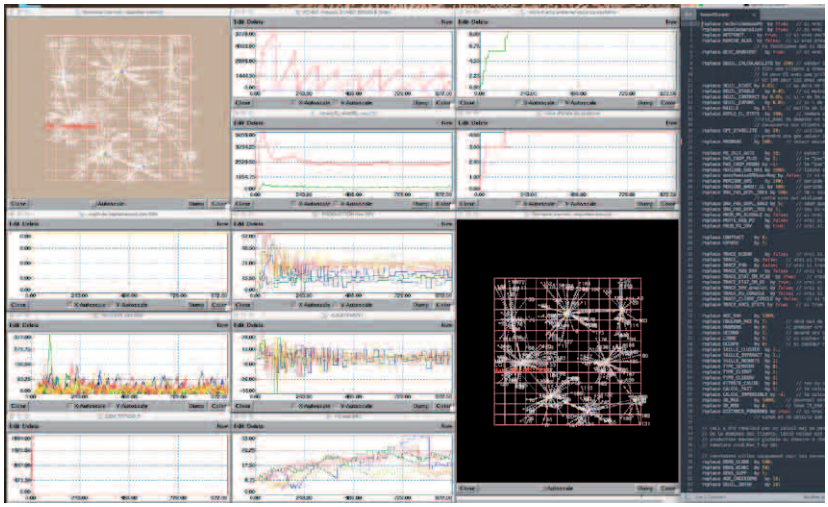


figure Fig-IV-E1-101

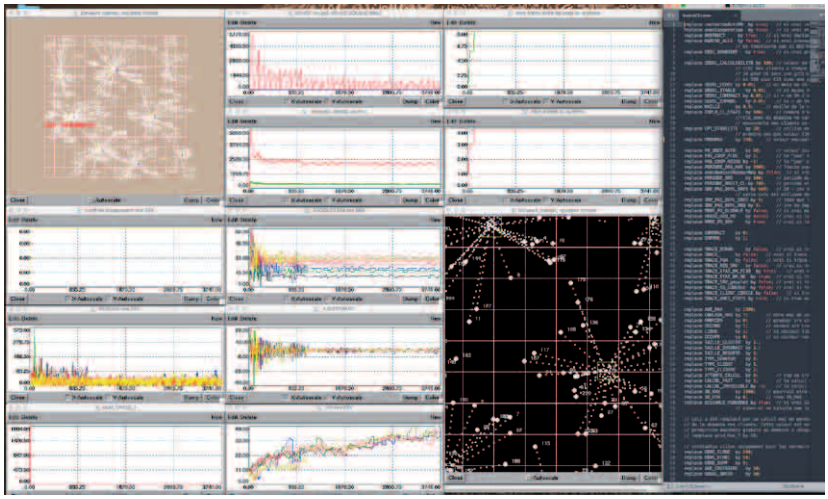


Fig-IV-E1-102

attracteur local; trop d'états signifie qu'il y a un risque d'oscillation dans une sorte de méta-attracteur local.
On comprend qu'il va falloir mesurer nos résultats et les comportements des heuristiques, à l'aune des solutions optimales, calculées.
Nous proposons deux extensions successives du code *oRis* de notre simulation, que nous appelerons respectivement la version v4 et v5.
Pour mieux comprendre la dynamique du système, l'idée de la v4 est de voir en temps réel comment les heuristiques fonctionnent vis-à-vis des optimaux locaux/calculés (CG^* local et propre à chaque région/serveur).
Pour chaque région, nous visualisons en temps réel (au fur et à mesure de l'exploration) son ou ses optimums locaux avec son rectangle de recouvrement associé. Ainsi, la comparaison entre les heuristiques se fait visuellement et en temps réel.

Extension du code de la simulation : version v4

Nous visons ici l'affichage en temps réel du changement des états dans le processus de résolution de la simulation. Pour cela nous visualisons l'évolution des diagrammes de Voronoï, tels que nous les avons introduits au chapitre III, dans la partie III-D2, lors de l'étude de la frontière entre *k-partitions*. Nous visualisons également les minima locaux des régions, afin de voir comment se comportent les serveurs dans ce processus de résolution.

Cette évolution de code passe par la création de deux classes (*CgReg* et *ZoneRec*) qui vont être utilisées par les serveurs, mais aussi de nouvelles méthodes des serveurs et des modifications dans le code des clients, des requêtes, des serveurs et du Manager.
La difficulté de ces modifications réside dans une bonne interaction (échange d'information) entre les différents acteurs en jeu, dont les différents scénarii, propres à chacun, sont assez subtils.

Ce que nous voyons sur la figure *Fig-IV-E1-100* est le résultat en temps réel des heuristiques résolutives. Chaque serveur est autonome et indépendant des autres serveurs, mais peut coopérer avec eux pour satisfaire, en premier lieu, ses propres intérêts, dans une résolution globale optimale (tous les serveurs doivent être satisfaits, mais aussi les clients, voire les responsables des infrastructures où se passent les échanges). Ce résultat se matérialise ici par la répartition (positions) des serveurs sur les clients, ces derniers étant fixes. Comme déjà indiqué, nous savons que le nombre de serveurs fait également partie de la solution. Sur la figure *Fig-IV-E1-100*, ce nombre est imposé.
L'idée simple, mais très efficace, est de calculer en temps «pseudo réel» (à la fin de chaque PO de chaque serveur) la position exacte de la solution (CG* local) optimale pour chaque région (une par serveur), et de l'afficher en temps réel sur la grille.
Comme nous allons le voir sur des exemples, le CG* local calculé sur la grille (en incluant la maille) pourra être moins bon (sous-optimal), bien qu'exact, que celui trouvé par les heuristiques. Cela sera d'autant plus fréquent que la maille (constante *MAILLE*) de la grille sera grande (tend vers 1). En effet les serveurs de la simulation se déplacent dans l'espace continu euclidien en deux dimension. En affinant la maille de la grille, les solutions exactes seront de plus en plus proches de l'optimum réel, mais plus coûteuses à calculer.
Cette approximation, des solutions locales exactes, reste néanmoins pertinente pour comprendre et suivre les heuristiques résolutives.
Chaque serveur crée une instance de la classe *CgReg*, pour y mémoriser la position (x/y) calculée de l'optimum local exact, ainsi que la valeur du coût (CG* local) correspondant. Chaque instance de *CgReg* crée, à son tour, une instance de la classe *ZoneRec*, pour y mémoriser les points de l'enveloppe (ici un rectangle) de recouvrement de la région correspondant au serveur courant. La constante globale *TRACE_CgReg* (issue du fichier «FusionV4.cstes») permet de lancer le calcul et la visualisation des positions exactes et des zones de recouvrement associées (dans le *main* de ZoneRec) quasiment en temps réel.

La mécanique globale est la suivante:

- dans le *main* de chaque serveur, en fin de chaque PO (période d'observation), propre à chaque serveur, le serveur courant regarde s'il doit détruire des objets *Interaction* qui caractérisent le lien (producteur/consommateur) qu'il entretient avec chacun de ses clients. Pour cela il fait appel à la méthode *del_interactions*, qui profite de la destruction d'une ou plusieurs interactions (départ de clients) pour recalculer les coordonnées ({(xdom,ydom),(Xdom,Ydom)}) du rectangle de recouvrement de la région qui lui est associée;
- à l'image des objets IPC «segment de mémoire» sur Linux[1], lorsque les requêtes voient un serveur cible (pour aller y satisfaire la demande de leur client),

1 Ayant longtemps enseigné, entre autres choses, les IPC sur UNIX, ce fut pour moi l'occasion de rendre hommage aux concepteurs de ce merveilleux système d'exploitation.

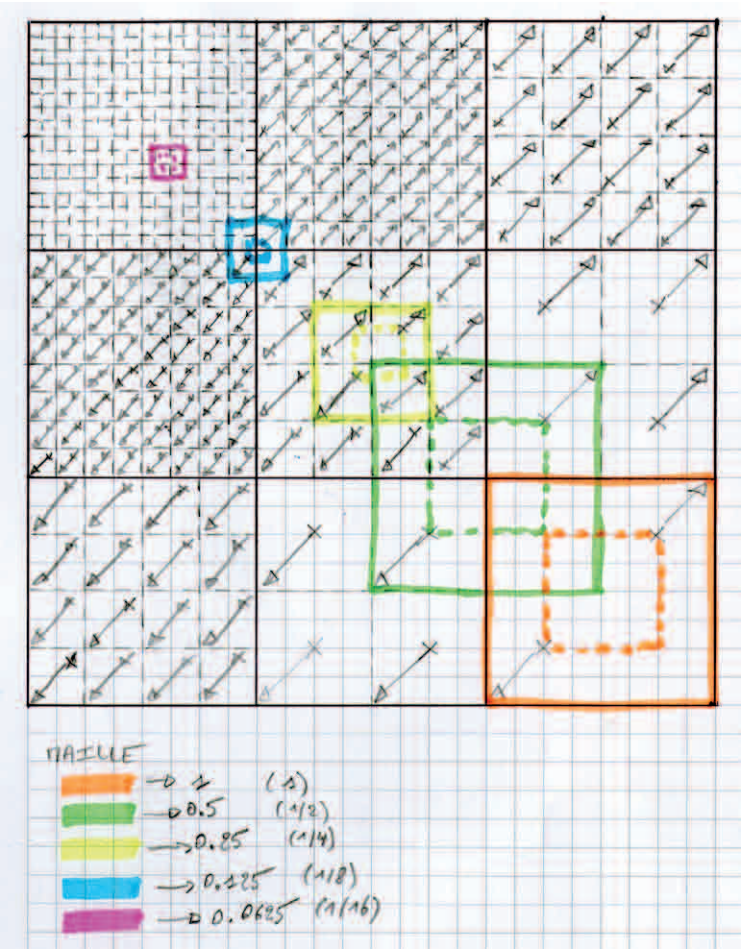


Fig-IV-E1-103

elles vérifient si ce serveur est identique à celui qu'elles voyaient au top précédent. Dans le cas contraire, cela signifie, soit que l'ancien serveur n'existe plus, soit qu'il est parti ailleurs, soit encore qu'un autre serveur se soit rapproché de la requête en question (les serveurs sont mobiles et autonomes). Dans pareil cas, la requête prévient cet ancien serveur que son client n'est plus en interaction avec lui. D'où l'idée de marquer «à détruire» (méthode *AD_interaction*) l'objet *Interaction* de cet ancien serveur, correspondant au client de la requête. Dans la mesure où l'objet *Interaction* du serveur ne sera détruit qu'à la fin de sa PO en cours, si entre temps (avant la fin de cette PO) l'ancien serveur redevient le serveur cible de la requête, alors l'objet *Interaction* en question sera simplement réactivé. C'est une sorte de fonction «cache». Cette fausse bonne idée introduit de gros problèmes de cohérence dans la simulation. On s'en apercevra plus tard (après un très long débogage du code) et nous devrons retirer ce mécanisme ;

- encore dans le *main* des serveurs, en fin de PO, chaque serveur lance sa méthode *calculeCgReg* (si la constante TRACE_CgReg vaut *true*) pour recalculer le CG* local et afficher la position associée et son rectangle de recouvrement. Pour cela cette méthode met à jour les coordonnées du rectangle ({(xdom,ydom),(Xdom,Ydom)}) non plus sur la base du départ ou de l'arrivée de nouveaux clients, mais en calant le rectangle sur la grille en tenant compte de la maille (*MAILLE*). En effet c'est sur cette base que le calcul exact pourra être effectué (nombre fini de points). Comme le montre la figure *Fig-IV-E1-103* nous voyons l'alignement du rectangle de recouvrement de chaque région sur la grille en fonction de la maille associée. Nous avons représenté les valeurs de maille allant de 1 à 0.0625 (1/16). Plusieurs stratégies peuvent être choisies. Nous avons retenu celle qui consiste à prendre la surface (trait plein) sur la grille immédiatement englobant le rectangle de recouvrement (trait pointillé) défini par les positions des clients. Cette approche assure de ne pas oublier de points, même si elle est pessimiste, puisqu'elle ajoute des points inutiles pour le calcul de CG* local. L'approche qui consisterait à prendre la surface immédiatement englobée par le rectangle de recouvrement pourrait conduire à des erreurs,

surtout pour une maille grande. Le code qui réalise ce recadrement sur la grille est le suivant :

```
for(i=1;i<(1/MAILLE);i++)  // maj xdom sur la sous-grille
    if((xdomBarre+(1-(i*MAILLE)))<xdom) xdom=xdomBarre+(1-(i*MAILLE));     // xdom=xdomBarre+0.5
    else if((xdomBarre+(1-(i*MAILLE)))>xdom) xdom=xdomBarre;               // xdom=xdomBarre
  for(i=1;i<(1/MAILLE);i++)  // maj de ydom sur la sous-grille
    if((ydomBarre+(1-(i*MAILLE)))<ydom) ydom=ydomBarre+(1-(i*MAILLE));     // ydom=ydomBarre+0.5
    else if((ydomBarre+(1-(i*MAILLE)))>ydom) ydom=ydomBarre;               // ydom=ydomBarre
  for(i=1;i<(1/MAILLE);i++)  // maj de Xdom sur la sous-grille
    if((Xdom-(1-(i*MAILLE)))>XdomBarre) Xdom=XdomBarre+(1-((i-1)*MAILLE)); // Xdom=XdomBarre+1
    else if((Xdom-(1-(i*MAILLE)))<XdomBarre) Xdom=XdomBarre+(1-(i*MAILLE));// Xdom=XdomBarre+0.5
  for(i=1;i<(1/MAILLE);i++)  // maj de Ydom sur la sous-grille
    if((Ydom-(1-(i*MAILLE)))>YdomBarre) Ydom=YdomBarre+(1-((i-1)*MAILLE)); // Ydom=YdomBarre+1
    else if((Ydom-(1-(i*MAILLE)))<YdomBarre) Ydom=YdomBarre+(1-(i*MAILLE));// Ydom=YdomBarre+0.5
```

Il suffit ensuite de calculer le coût (CG) pour chacun des points obtenus par le recadrage du rectangle de recouvrement ;

- la dernière étape, de cette mécanique globale, consiste, pour les requêtes qui arrivent à proximité (seuil) des serveurs visés, à leur demander s'ils peuvent assumer les demandes de leur client (méthode *Serveur::lancer_calcul*). Si le serveur visé est en capacité de traiter cette demande (ses ressources disponibles sont suffisantes) alors la méthode *Serveur::maj_interaction* est lancée avec une valeur de demande positive. Dans le cas contraire (les ressources du serveur sont insuffisantes par rapport à la demande), la même méthode *Serveur::maj_interaction* est appelée, mais avec, pour demande, une valeur négative. Cette méthode arrête le déplacement du serveur quand ce dernier est en mode expansion. Si la requête courante vient directement d'un client qui est nouveau pour le serveur, son rôle est de créer un nouvel objet *Interaction* dans les variables de classe du serveur, sinon de mettre à jour l'objet *Interaction* correspondant à ce client (c.-à-d. appartenant à la région associée au serveur). Cela exclut les requêtes résiduelles, celles qui ont échoué sur le serveur précédent et qui continuent leur recherche d'un autre serveur. Dans le cas de l'arrivée d'un nouveau client, si le serveur précédemment vu par ce client n'est pas nul ((Serveur) NONE), il faut prévenir (méthode *Serveur::AD_interaction*) ce serveur qu'il aura potentiellement un objet *Interaction* à détruire à la fin de sa PO courante. En effet ce client vient de changer de région. Une mise à jour du rectangle de recouvrement ({(xdom,ydom),(Xdom,Ydom)}) du serveur est donc lancée pour tenir compte de l'arrivée de ce nouveau client.

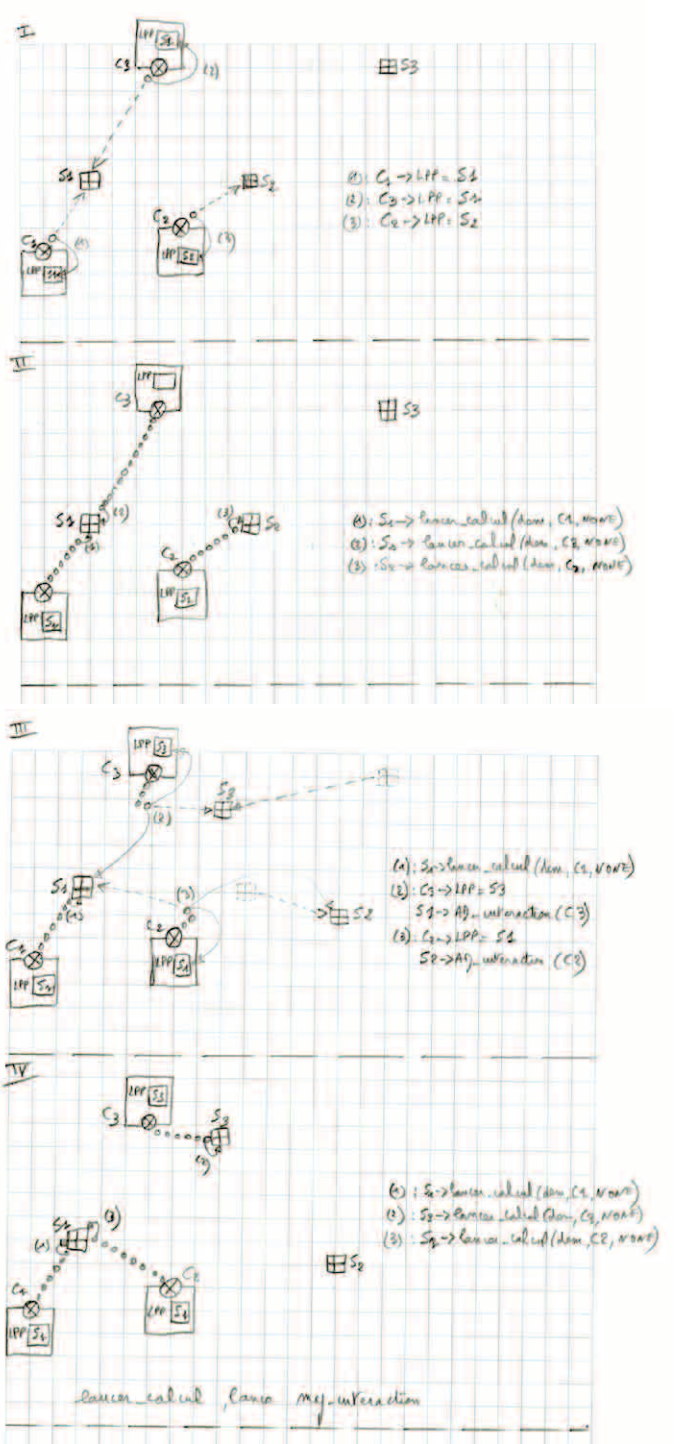
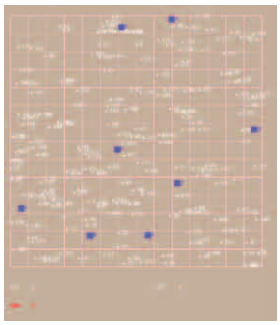
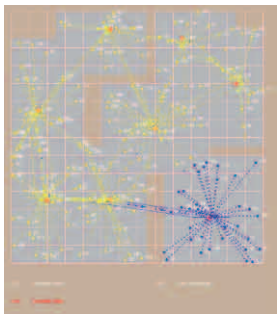
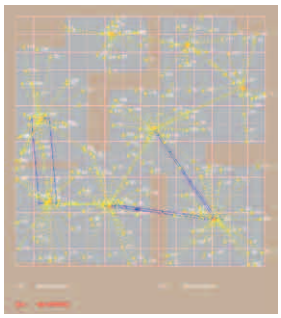


Fig-IV-E1-104/105

configuration n1
Serveur (x, y)
0.609106540032,3.25361832197
13.5631372703,7.64810204583
7.6510842087,1.77229559597
8.97208150149,13.7737955348
6.18155256481,13.3539567587
5.95124305317,6.54199458963
9.30306796697,4.66332088349
4.43408875094,1.72963712445







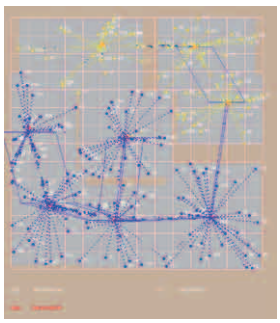


Fig-IV-E1-106

Le scénario présenté sur les figures *Fig-IV-E1-104/105* illustre l'usage des méthodes de mise à jour pour le calcul en temps réel des enveloppes de recouvrement des serveurs.
Nous avons trois serveurs (S_1,S_2,S_3), trois clients (C_1,C_2,C_3) et quatre étapes (I,II,III,IV). Dans l'étape I et II les clients lancent leurs requêtes qui recherchent les serveurs les plus proches (S_1 pour C_1 et C_3, S_2 pour C_3). À l'étape III le serveur S_3 se rapproche de C_3 et le serveur S_2 s'éloigne de C_2.
Une réorganisation en temps réel des requêtes s'opère avec comme résultat un état stabilisé à l'étape IV.

Regardons, à partir de l'exemple $C_{200}S_8$ vu précédemment, le résultat de ces modifications (FusionV4) sur l'interface de suivi de la simulation.
Nous allons voir que ces résultats nous offrent des perspectives qualitatives, mais aussi quantitatives de compréhension des solutions proposées par les heuristiques résolutives.

Nous avons regardé $C_{200}S_8$ en le lançant trois fois avec respectivement une maille à ½, ⅛ et 1/16 (cf. la figure *Fig-IV-E1-106*). Grâce à un dispositif, généré automatiquement, de sauvegarde sur fichier des conditions initiales (positions initiales des clients et des serveurs) du problème C_SS_J, nous pouvons comparer ces trois exécutions successives.

Voici les résulats obtenus :

MAILLE	½	⅛	0.0625
nbre de cycle	766	777	212
CG^* (exact)	367.183197889	365.259234233	363.03821171
CG (heuristique)	373.562971889	389.476355287	391.22145146

En réduisant la maille on augmente le nombre de points à calculer pour obtenir les coûts optimaux locaux (CG^*) de chaque région. Sur notre grille de deux-cent-vingt-cinq (15x15) points avec une maille MAILLE=0.0625, nous avons entre douze-mille et quatorze-mille points à calculer. Ceci ralentit considérablement le déroulement de la simulation. C'est ce qui explique le nombre de deux-cent-douze cycles, très inférieur aux autres. Nous touchons la limite de l'approche exacte, contrairement à l'approche heuristique qui n'a pas cette limite, tout en en ayant d'autres.
Conformément à ce que l'on pouvait attendre, le tableau ci-dessus nous montre bien (3e ligne) que plus la maille est fine et plus la valeur du coût global (CG^* (exact)) se rapproche de l'optimum absolu.

Là encore, seule l'approche heuristique est capable de s'approcher davantage de l'optimum global exact, dans la mesure où elle travaille en continu au niveau du déplacement des serveurs.
En revanche nous verrons par la suite qu'il n'est pas aussi simple de trouver cet optimum, en déjouant tous les pièges tendus par les innombrables optima locaux. C'est ce qui explique les valeurs trouvées par les heuristiques dans le tableau ci-dessus, qui ne sont pas très proches de l'optimum.

La nouvelle interface (v4) affiche en temps réel l'évolution du coût global (CG), courbe rouge sur la figure *Fig-IV-E1-107*, calculé par les heuristiques; elle affiche aussi le calcul exact du coût global optimal (somme des CG* de chaque région), courbe verte sur la figure *Fig-IV-E1-107*. On voit sur cet exemple que le coût des heuristiques suit la courbe de celle des coûts exacts calculés, avec un léger décalage.

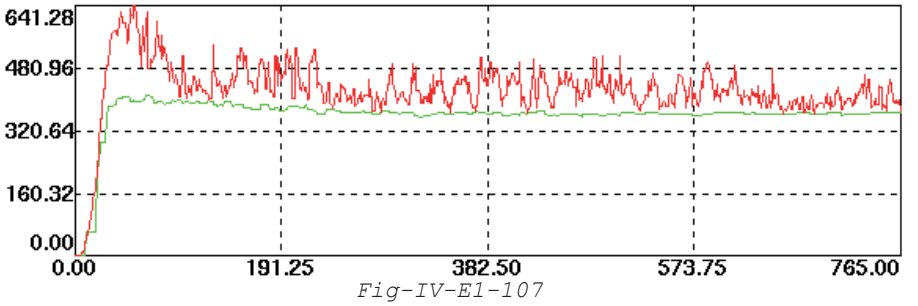


Fig-IV-E1-107

Sur la grille, de la figure *Fig-IV-E1-108a*, les deux valeurs de la figure *Fig-IV-E1-107* (CG/CG*) s'affichent aussi en temps réel (sur le bas de l'écran).
Nous avons également les positions des CG*, locaux à chaque région (une par serveur), sous la forme d'un losange plein, de couleur orange. Le rectangle de recouvrement de chaque région apparaît en fond gris clair. C'est dans ces zones que sont réalisés les calculs exacts alignés sur la grille et sa maille.
On voit sur cet exemple que les serveurs sont tous positionnés non loin des losanges propres à leur région, donc a proximité de l'optimum local. Les losanges permettent de voir comment se comportent les serveurs lors de leurs déplacements pendant la résolution.

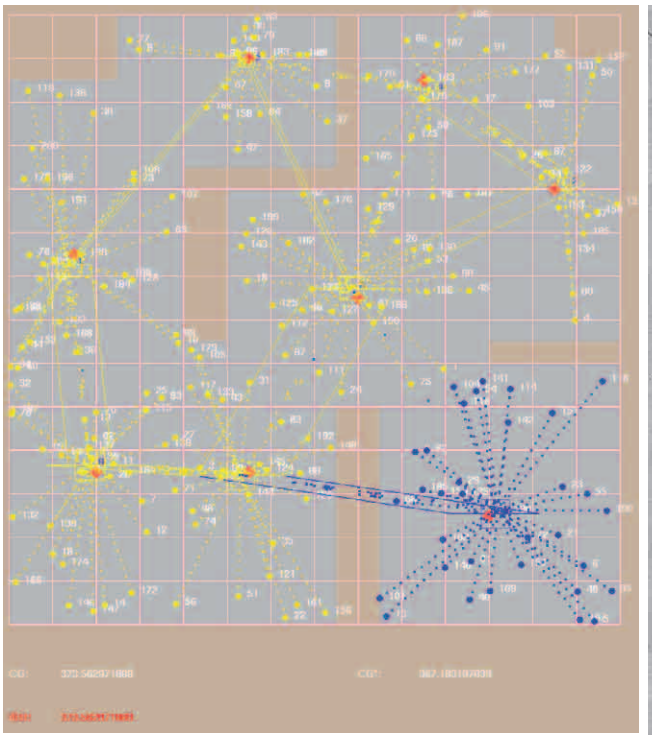


Fig-IV-E1-108a

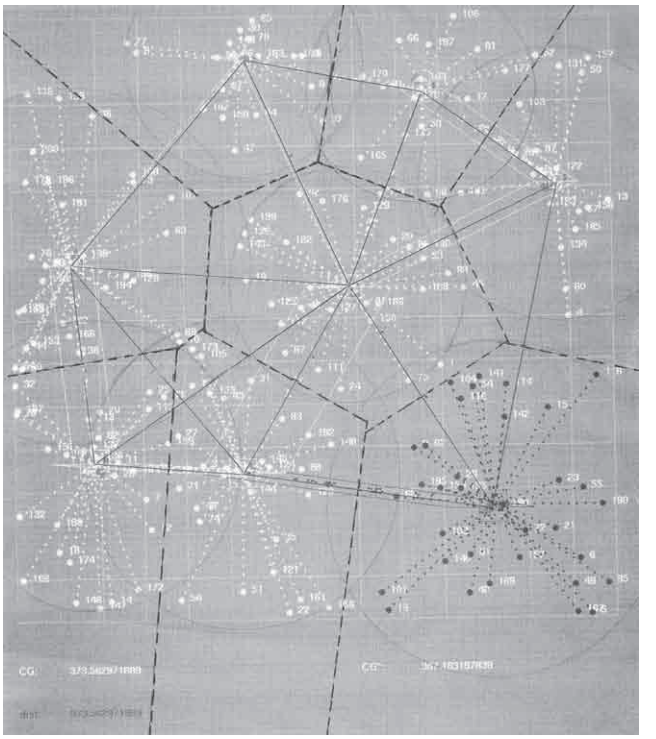


Fig-IV-E1-108b

Nous avons dessiné, sur la figure *Fig-IV-E1-108b*, le diagramme de Voronoï (en traits pointillés) et la triangulation de Delaunay (en traits pleins) correspondant à l'état atteint par la résolution sur cet exemple.

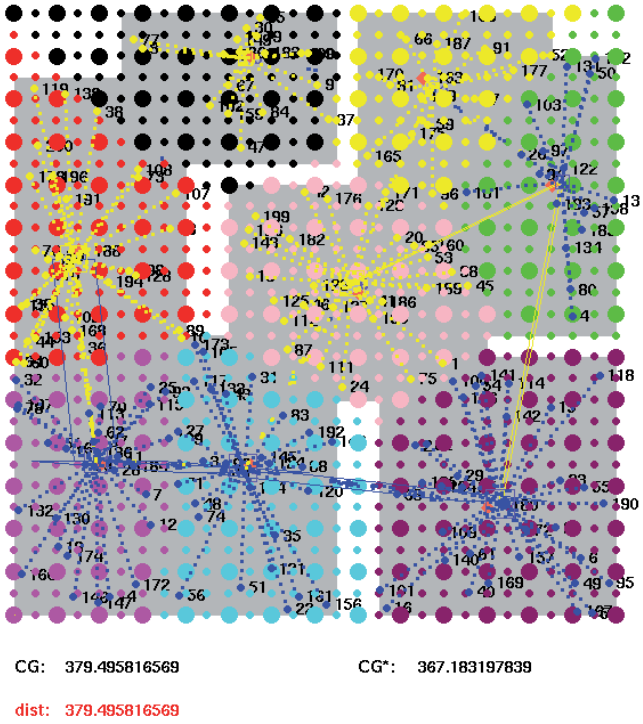


Fig-IV-E1-109

Pour mieux visualiser, pendant la résolution, l'évolution du diagramme de Voronoï, et de la triangulation de Delaunay associée, nous avons coloré, région par région (cf. la figure *Fig-IV-E1-109*), les points de la grille (gros cercles pour les unités et petits cercles pour les points de la maille) avec la couleur associé à chaque serveur, représentant ces régions. Pour mémoire, un état du système (résolution partielle) correspond, pour un ensemble de clients fixes, à un ensemble de serveurs positionnés sur la grille. Chacun définit alors une région ou zone du diagramme de Voronoï.

L'indicateur de qualité de la solution heuristique, consistant à tendre vers la solution optimale, peut être à la fois visuel et quantitatif.
En effet la surface totale de la grille non recouverte par une région sera d'autant plus grande que la solution sera stable.

Deux-cents clients et huit serveurs $(C_{200}S_8)^{++}$

Pour appréhender le problème de la résolution heuristique de C_iS_j avec un grand nombre de clients et un nombre relativement conséquent de serveurs, nous allons continuer avec l'exemple précédent (deux-cents clients et huit serveurs) sur la v4.

Nous nous intéressons à l'impact de la position initiale des serveurs sur les résultats de la résolution. Nous avons retenu huit configurations du positionnement initial des serveurs, dont les cinq premières (n°1 à n°5) sont aléatoires, alors que les trois autres (n°6, n°7 et n°8) sont très singulières (choix à dessein). En effet dans la n°7 tous les serveurs sont exactement au centre de la grille (7,7), dans la n°6 quatre serveurs sont regroupés dans le coin haut gauche de la grille, les quatre autres dans le coin bas droit, et dans la n°8 quatre serveurs occupent la diagonale de la grille, les quatre autres sont répartis de façon symétrique de part et d'autre de cette diagonale.
Cet exemple, bien que très raisonnable en nombre de serveurs, nécessite des calculs conséquents si l'on veut connaître le nombre de cas à parcourir pour obtenir l'optimum global CG^* exact (par rapport à la grille). Si l'on se réfère à la formule *(7)*, vue au chapitre III, ce nombre de cas s'élève sur une grille de deux-cent-vingt-cinq (15x15) points, avec une maille de 1, à 143 642 651 595 300 $((15^2!)/(8!(15^2-8)!)$, soit 143 642 milliards.
On voit tout de suite que la confrontation des heuristiques aux calculs exacts légitime les heuristiques, tout en compliquant cette légitimité.
Dans la suite de cette partie nous reviendrons sur des exemples «significatifs» et calculables, afin de garder cette confrontation «heuristiques/calcul exact»

le plus longtemps possible dans nos analyses. Nous prendrons des exemples avec deux, trois et quatre serveurs. Cette étude sommaire nous donnera une idée précise de la problématique des grands systèmes (grand nombre de clients mais aussi de serveurs).

Regardons l'impact de la position initiale des serveurs sur le processus de résolution, mais aussi celui de la stratégie de réorganisation. Rappelons que la méthode *Serveur::restructurer* conduit à définir une technique de réorganisation et une seule au temps 't' (concomitante pour tous les serveurs), parmi la marche aléatoire, la descente de gradient ou l'algorithme de contraction/expansion (C/E).
Nous regarderons également le rôle de la vitesse de déplacement des serveurs, qui est directement liée à la notion d'exploration du nombre d'états du système, lors de la résolution.

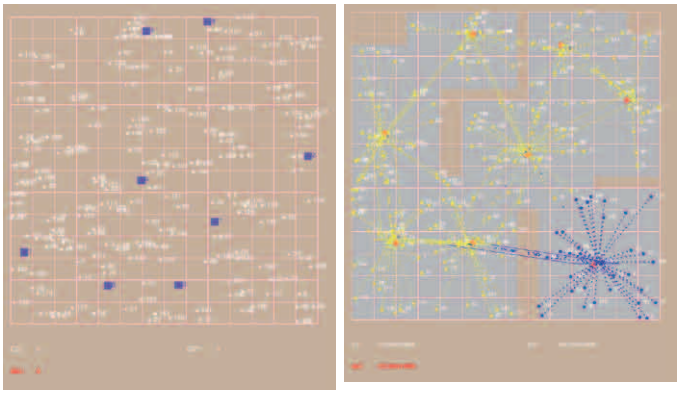


Fig-IV-E1-110 a/i (n°1)

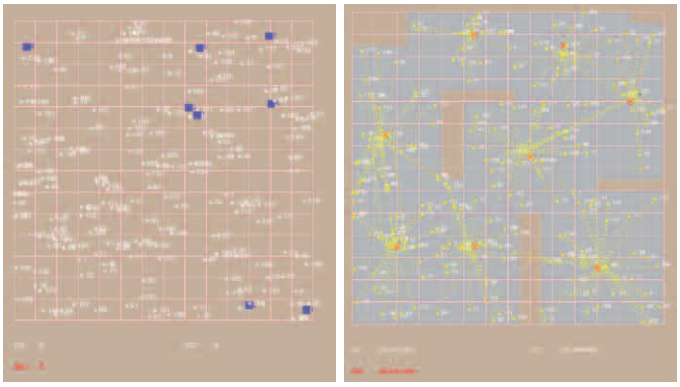


Fig-IV-E1-110 c/k (n°3)

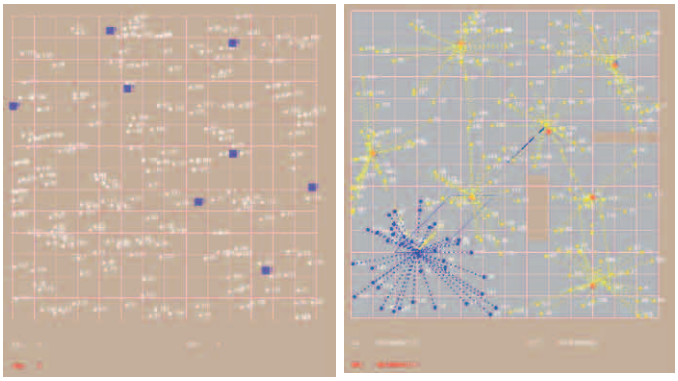


Fig-IV-E1-110 e/m (n°5)

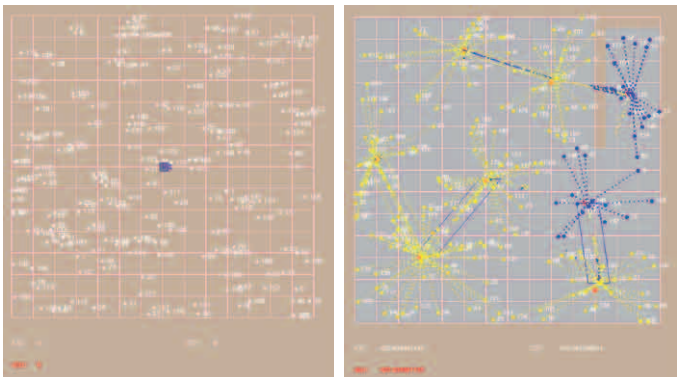


Fig-IV-E1-110 g/o (n°7)

Les huit serveurs ont les positions initiales (n°1 à n°8) que l'on peut voir (carrés de couleur bleu) sur les figures *Fig-IV-E1-110a/…/h*.
Les figures *Fig-IV-E1-110i/…/p* nous montrent l'état de la simulation au bout d'un nombre suffisant de tops d'horloge (dernière colonne du tableau ci-dessous) pour atteindre un état stable.
Voici le tableau des résultats obtenus avec ces huit configurations (n°1 à n°8) :

configurations (n1-n8)	**tops**	$CG^*_{200,8}$**/CG**	**nb états**
(a)-i	766	367/373	3
(b)-j	772	365/369	2
(c)-k	802	366/390	3
(d)-l	780	376/404	1
(e)-m	776	375/395	5
(f)-n	2174	365/371	4
(g)-o	783	375/427	8
(h)-p	955	385/430	2

Sur le plan analytique, ce tableau nous permet de tirer plusieurs enseignements. On voit que les résultats obtenus, où $CG^*_{200,8}$ est la valeur optimale calculée sur la grille (incluant sa maille) de l'état en cours et CG la valeur obtenue pour ce même

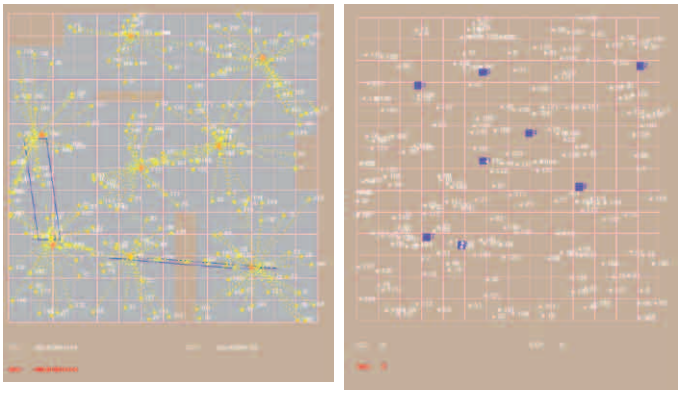


Fig-IV-E1-110 j/b (n°2)

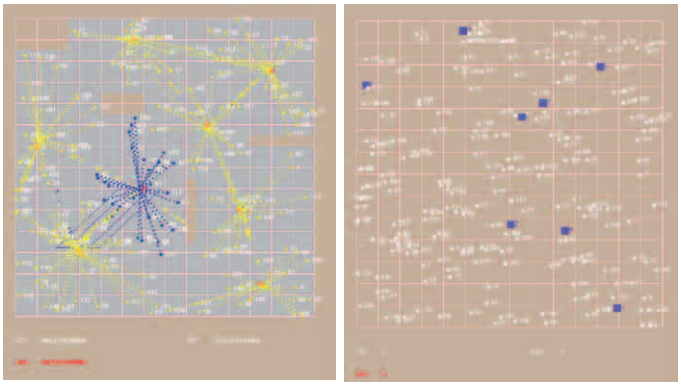


Fig-IV-E1-110 l/d (n°4)

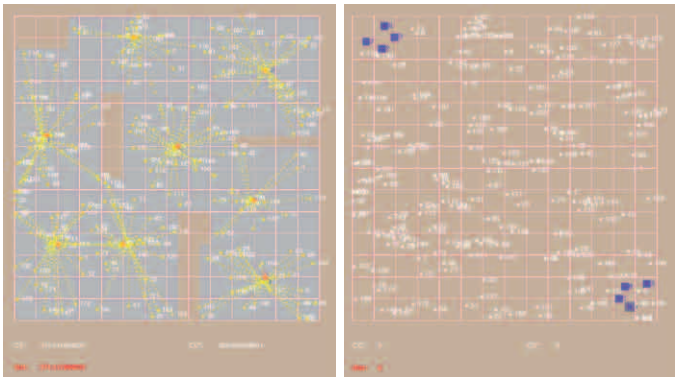


Fig-IV-E1-110 n/f (n°6)

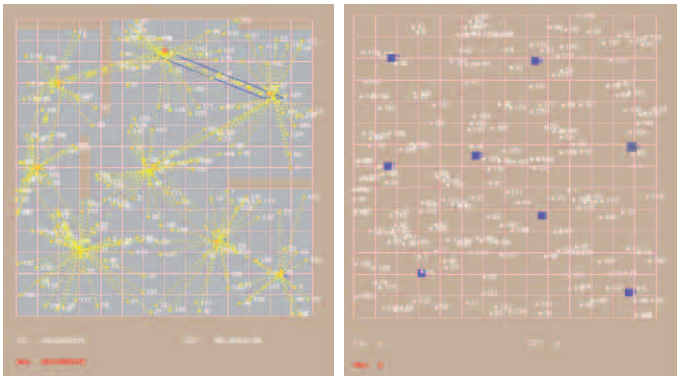


Fig-IV-E1-110 p/h (n°8)

état par la simulation, n'ont pas une très grande variabilité les uns par rapport aux autres. Même s'il est difficile de calculer l'optimal absolu (CG^* avec huit régions), on peut voir que les états retenus varient dans l'intervalle [365,385] et sont probablement en dehors de cet optimum absolu. Ceci corrobore les données du nombre d'états (4e colonne). En effet les états parcourus sont très peu nombreux, alors que le nombre d'états possibles est faramineux (143 642 milliards pour une maille de 1).
Cela pose immédiatement le problème des heuristiques pour des situations où le nombres de protagonistes (en particulier les serveurs) est réaliste, c.-à-d. important. On retombe sur la même difficulté/complexité que pour les calculs exacts, dans pareille situation, mais une difficulté/complexité d'une autre nature. Elle n'est plus calculatoire mais exploratoire. Comment casser la complexité de l'exploration d'un nombre extraordinaire de situations ?

Avant de poursuivre nos études et tirer les conséquences qui s'imposent pour dépasser ces difficultés, nous pouvons encore commenter ces premiers résultats.
Cette fois ce ne sera plus sur le plan analytique, mais sur la géométrie/topologie des solutions trouvées.
En observant ces dernières nous voyons se dessiner plusieurs configurations, dont certaines sont récurrentes. C'est le cas par exemple des propositions 'i', 'j', 'k' et 'n' qui sont très voisines en terme de placement des serveurs, donc qui partagent le même état. Le calcul des CG^* associé confirme cette observation, puisque les valeurs sont très proches ($CG^*_{200,8} \in [365\text{-}367]$). Il en est de même pour les propositions 'l', 'm' et 'o' ($CG^*_{200,8} \in [375\text{-}376]$). Seule la proposition 'p' est à part ($CG^*_{200,8}=385$).

On voit encore que la topologie des états représente un autre moyen d'expression et donc de raisonnement ou de prise en compte dans le fonctionnement et l'analyse de nos simulations.

Nous avons fait d'autres expériences, toujours à partir de cet exemple ($c_{200}s_8$) dans la configuration n°2 (cf. la figure *Fig-IV-E1-110 j/b*) sur une grille de deux-cent-vingt-cinq (15x15) points avec une maille de ½. Nous avons utilisé toutes les possibilités de configuration comportementale de notre simulation.
Nous avons regardé l'impact du comportement de réorganisation choisi, parmi les combinaisons possibles (RESTRUCT, ALEA, GRADIENT, ou RESTRUCT+GRADIENT). Nous avons aussi regardé les résultats en utilisant ou non le calcul automatique de la période d'observation (PO) des serveurs. Enfin nous avons fait varier la vitesse de déplacement des serveurs entre une unité de distance par top logique (vitesse la plus rapide) et 1/500 d'unité de distance par top logique (vitesse beaucoup plus lente).

Le tableau suivant montre les résultats obtenus en précisant certains paramètres significatifs :

n°2	top	Auto PO	avec Coop	Restruct	Gradient	Alea	Vitesse srv (V)	nbr états	$CG^*_{200,8}$	$CG_{200,8}$
n2.1	727	oui	non	oui	oui	non	1/500	2	360	407
n2.2	728	oui	oui	oui	non	non	1/500	1	372	425
n2.3	724	oui	oui	oui	oui	non	1/500	2	368	376
n2.3	2283	oui	oui	oui	oui	non	1/500	2	368	376
n2.4	1471	oui	oui	non	non	oui	1/500	1	371	385
n2.5	1504	oui	oui	non	oui	non	1/3	2	368	371
n2.5	3292	oui	oui	non	oui	non	1	2	368	371
n2.6	1614	oui	oui	oui	non	non	1/3	28	416	583
n2.7	4185	non	non	oui	oui	non	1/500	2	368	453

La configuration n2.1 confirme, avec son $CG^*_{200,8}$=360, que nous n'avions pas parcouru l'état global optimal avec les configurations n°1 à n°8 (les meilleures, n°2 et n°6, passaient par $CG^*_{200,8}$=365).
Globalement ce tableau nous montre que le choix des comportements possibles et synchronisés entre les serveurs, ne conduit pas à une solution globale intéressante. Nous peinons à rencontrer la solution optimale, ce qui s'explique par le fait que le système de résolution n'explore pas assez d'états. Ce n'est pas la seule raison.

En effet le fait, que tous les serveurs adoptent toujours le même comportement simultanément pendant toute la résolution, même lorsqu'on combine l'algorithme de contraction/expansion avec la descente de gradient, produit une résolution qui n'exploite pas assez les propriétés de l'espace des états.

Si l'on se réfère aux nuées d'oiseaux artefactuelles et aux travaux effectués dans le cadre des thèses d'Hugo Pommier[2] et de Benoît Romito[3], chaque entité de la nuée dispose d'un ensemble de comportements élémentaires qu'elle pourra utiliser selon son bon vouloir, dans la mesure où elle respectera les règles (celles de Reynolds) d'appartenance et de co-construction du groupe auquel elle appartient.

2 « Placement et stockage de l'information bio-inspiré : une approche orientée agents mobiles. », thèse de doctorat de l'Université de Caen Basse-Normandie, en informatique, 2010.
3 « Stockage décentralisé adaptatif : autonomie et mobilité des données dans les réseaux pair-à-pair. », thèse de doctorat de l'Université de Caen Basse-Normandie, en informatique, 2012.

Autrement dit, les membres d'une nuée utilisent des comportements élémentaires de façon désynchronisée. Dans une vraie nuée d'oiseaux, un oiseau peut voler dans une direction, alors qu'un autre en choisit une autre…, dans la limite de la déformation de la nuée. Lorsqu'un avion se déplace dans une certaine direction, à une vitesse donnée, rien n'empêche les passagers, sauf à des moments particuliers (décollage, atterrissage, turbulence), de se déplacer dans la direction et à la vitesse qu'ils veulent, dans la limite de l'espace intérieur de l'avion.

En revenant au tableau ci-dessus, nous observons qu'une vitesse de déplacement rapide (cas où V=⅓) peut, dans des situations particulières, comme par exemple la configuration n2.6, provoquer une exploration plus vaste de l'espace des solutions en nombre d'états rencontrés (vingt-huit pour n2.6).
Nous verrons, dans notre étude (cf. la partie IV-E2), qu'une augmentation de la vitesse de déplacement peut conduire à ne jamais rencontrer certains états, qui peuvent, dans certains cas malheureux, être ceux que l'on recherche; et ce, même si on va parcourir/traverser plus d'états qu'avec un déplacement moins rapide…
Nous tirons deux conclusions de ces premiers résultats qui nous conduisent vers une refonte complète du moteur de la simulation.
Nous étions passé, avec la version v3, à côté du problème de la synchronisation fonctionnelle des serveurs (que l'on trouve avec les oiseaux dans les nuées réelles) en travaillant sur des systèmes réduits en nombre d'éléments. La confrontation à des systèmes plus grands a pointé la complexité exploratoire des systèmes «naturels», où l'asynchronisme semble jouer un rôle important, ainsi que la vitesse de déplacement des serveurs. À la complexité calculatoire s'oppose frontalement pour nous cette complexité exploratoire.
C'est l'objectif de la version v5, puis de la v6 d'y remédier.

Extension du code de la simulation : version v5

La principale évolution du code consiste à désynchroniser les serveurs entre eux, dans le sens où chaque serveur disposera du graphe d'états des comportements possibles et pourra modifier son comportement, conformément à ce graphe partagé par tous les serveurs. L'idée est, sur le plan comportemental, de rendre les serveurs autonomes les uns par rapport aux autres, avec des possibilités de synchronisation entre eux, permettant d'accéder à une résolution globale du problème posé.
Cela touche principalement le code des méthodes *Serveur::restructurer* et *Serveur::main*. D'autres modifications du code découleront de la mise en place de ce nouveau graphe d'états.

Revenons rapidement sur le fonctionnement jusqu'à la v4. Au départ nous avons une situation initiale (positionnement x/y) dans laquelle des serveurs sont au milieu des clients. Dans la mesure où les clients diligentent des requêtes pour

trouver les serveurs «au plus proche», cette situation correspond à l'état initial de la résolution avec ses régions correspondantes. À chaque top de la simulation, hors top correspondant à celui de la fin de la période d'observation (PO) en cours, les serveurs sont soit en recherche d'une direction aléatoire (mode ALEA), soit en descente de gradient (mode GRADIENT), soit en recherche d'une direction suivant l'algorithme de contraction/expansion (mode RESTRUCT), soit une combinaison de l'algorithme de contraction/expansion et de celui de descente de gradient. Le choix du mode est fait une fois pour toute en début de simulation et pour tous les serveurs. En dehors de la recherche de gradient qui consiste, à chaque top, à trouver la position suivante des serveurs en suivant le gradient (CG↓) et à y aller (méthode *Serveur::calculCible_allerY*), les autres méthodes consistent à calculer une cible (x/y_cible) en fin de PO et à s'en rapprocher (méthode *Serveur::aller_vers*) à chaque top ordinaire (≠ de celui de fin de PO).
En fin de PO, les serveurs refont un point (méthode *Serveur::restructurer*) sur la cible à suivre, sauf s'ils sont en descente de gradient. Ils profitent également de ce moment privilégié pour mettre à jour les objets *Interaction* (méthode *Serveur::del_interactions*) en détruisant ceux qui correspondent à des clients qui ont changé de serveur.

La méthode *Serveur::restructurer* fonctionne de la façon suivante :

- mode ALEA : le serveur explore (calcul d'une direction à prendre via une position cible) l'enveloppe de recouvrement (définie par x/y_dom et X/Y_dom) de la région à laquelle il appartient et qu'il définit. Dans ce cas le point cible est arrondi sur la grille. Une autre méthode consiste à prendre pour cible l'un des clients de la région pris au hasard ;
- mode GRADIENT : les serveurs ne font rien ;
- mode RESTRUCT : l'algorithme de contraction/expansion trouve la nouvelle direction à prendre pour la PO à venir, conformément à ce qui se faisait dans la v3.

La méthode *Serveur::restructurer* de la v5 a été profondément modifiée. Nous avons implanté, dans la méthode *Serveur::main*, un nouveau graphe d'états des comportements des serveurs, très complet. Chaque serveur possède une variable de classe *monMode* qui détermine son mode courant à chaque top. De cette façon des serveurs différents peuvent être simultanément dans des modes différents, l'un pouvant être dans une descente de gradient, alors qu'un autre pourra être en mode recherche aléatoire.
Les transitions, pour passer d'un mode à un autre, sont codées dans la méthode *Serveur::main*, dans sa partie top ordinaire (≠ fin de PO). Avant d'aborder le graphe des états possibles du comportement des serveurs, revenons sur la principale nouveauté de la méthode *Serveur::restructurer*. Il s'agit de la direction à prendre en mode aléatoire.

Dans ce mode aléatoire il faut regarder s'il faut calculer une nouvelle direction (x/y_cible). L'objectif de ce mode est d'explorer l'espace des états du système, c.-à-d. trouver un nouvel état à partir de l'état courant.

Cette dernière contrainte permet de rester sur le graphe des états (sans discontinuité/rupture dans l'exploration). On a au moins deux solutions :

- la première consiste à prendre au hasard un point du rectangle de recouvrement de la région courante (*cls_dom*), qui devient la cible ;
- la seconde consiste à prendre au hasard l'un des clients de la région et à prendre comme cible l'intersection entre l'axe défini par la position actuelle du serveur et ce client, avec la frontière externe de l'enveloppe de recouvrement.

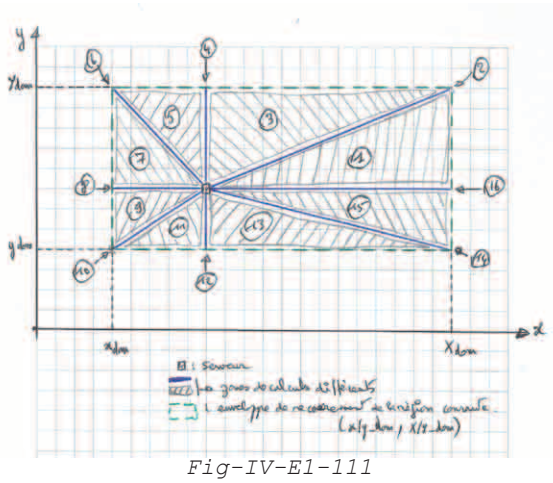


Fig-IV-E1-111

La première solution est simple à mettre en œuvre, car elle ne demande aucun calcul spécifique. La seconde est plus complexe et donc coûteuse en temps comme le montre le code ci-après et la figure *Fig-IV-E1-111* où l'on voit tous les cas à prendre en compte dans le calcul.

```
//************************************* à partir d'un client ********************
int alea=floor(nbclvus*?);  // tirage aléatoire entre 0 et nbclvus-1 ; ? pour un float
                            // pris au hasard entre [0,1]
xcib=clvus[alea]->getX();xsrv=getX(); // de cette cible intermédiaire il faut aller
ycib=clvus[alea]->getY();ysrv=getY(); // jusqu'à l'enveloppe de recouvrement du srv
float tgA,tgA1,tgA2,tgA3,tgA4;  // tangentes des angles dans les quatre coins du
tgA=(ycib-ysrv)/(xcib-xsrv);   // rectangle de recouvrement
if((ycib>ysrv)&&(xcib>xsrv)) {
  tgA1=(Ydom-ysrv)/(Xdom-xsrv);
  if(tgA<tgA1) {x_cible=Xdom;y_cible=(1/(xcib-xsrv))*
                          (((ycib-ysrv)*Xdom)+((xcib*ysrv)-(ycib*xsrv)));}  // cas 1
  else if(tgA==tgA1) {x_cible=Xdom;y_cible=Ydom;}                            // cas 2
       else {x_cible=(1/(ycib-ysrv))*
             (((xcib-xsrv)*Ydom)+((ycib*xsrv)-(xcib*ysrv)));y_cible=Ydom;}  // cas 3
}
else
  if((xcib-xsrv)==0)
    if((ycib-ysrv)>0) {x_cible=xsrv;y_cible=Ydom;}                          // cas 4
    else {x_cible=xsrv;y_cible=ydom;}                       // (ycib-ysrv)<0, cas 12
  else
    if((ycib>ysrv)&&(xcib<xsrv)) {
      tgA2=(xsrv-xdom)/(ysrv-Ydom);
      if(tgA<tgA2) {x_cible=(1/(ycib-ysrv))*
             (((xcib-xsrv)*Ydom)+((ycib*xsrv)-(xcib*ysrv)));y_cible=ydom;}  // cas 5
      else if(tgA==tgA2) {x_cible=xdom;y_cible=Ydom;}                       // cas 6
           else {x_cible=xdom;y_cible=(1/(xcib-xsrv))*
                          (((ycib-ysrv)*xdom)+((xcib*ysrv)-(ycib*xsrv)));}  // cas 7
    }
    else
      if((ycib-ysrv)==0)
        if((xcib-xsrv)<0) {x_cible=xdom;y_cible=ysrv;}                      // cas 8
        else {x_cible=Xdom;y_cible=ysrv;}                                   // cas 16
      else
        if((xcib<xsrv)&&(ycib<ysrv)) {
          tgA3=(ydom-ysrv)/(xdom-xsrv);
          if(tgA<tgA3) {x_cible=xdom;y_cible=y_cible=(1/(xcib-xsrv))*
                          (((ycib-ysrv)*xdom)+((xcib*ysrv)-(ycib*xsrv)));} // cas 9
          else if(tgA==tgA3) {x_cible=xdom;y_cible=ydom;}                   // cas 10
               else {x_cible=(1/(ycib-ysrv))*(((xcib-xsrv)*ydom)+
                               ((ycib*xsrv)-(xcib*ysrv)));y_cible=ydom;}    // cas 11
        }
        else
          if((xcib<xsrv)&&(ycib<ysrv)) {
            tgA4=(xsrv-Xdom)/(ysrv-ydom);
            if(tgA<tgA4) {x_cible=(1/(ycib-ysrv))*
              (((xcib-xsrv)*ydom)+((ycib*xsrv)-(xcib*ysrv)));y_cible=ydom;} // cas 13
            else if(tgA==tgA4) {x_cible=Xdom;y_cible=ydom;}                 // cas 14
                 else {x_cible=Xdom;y_cible=(1/(xcib-xsrv))*
                         (((ycib-ysrv)*Xdom)+((xcib*ysrv)-(ycib*xsrv)));}  // cas 15
          }
  //*****************************************************************************
```

Regardons le graphe des états comportementaux des serveurs. Comme nous pouvons le voir sur la figure *Fig-IV-E1-112*, cette première version du graphe comporte seize transitions (N1 à N15, plus N5b) et huit états (RESTRUCT, GRADIENT, ALEApo, ALEA, DUPLICATION, EQUILIBRE, GRADIENT_2 et AUTOPOIESE).

Détaillons chacune de ces seize transitions:

- N1: le serveur courant est en mode RESTRUCT et y reste en appliquant l'algorithme de contraction/expansion (déplacement grâce à la méthode *Serveur::aller_vers* qui utilise la cible calculée par l'algorithme en fin de PO précédente). Le compteur *cpt_explo_etats1*, local au serveur courant, est incrémenté tant que le système n'explore pas un nouvel état et que le serveur reste en mode RESTRUCT. Si un nouvel état est exploré, ce compteur passe à zéro. Si ce compteur dépasse le SEUIL (EXPLO_CL_ETATS1) alors le serveur passe (N2) en mode GRADIENT. L'idée est de rester suffisamment longtemps dans ce mode pour que l'algorithme positionne le système sur un état intéressant en éliminant des états sous-optimaux;
- N2: l'algorithme de contraction/expansion ne permet pas de trouver de nouvel état, le serveur va passer dans le mode GRADIENT pour consolider la dernière solution trouvée;
- N11: le serveur courant est interrompu pendant sa descente de gradient par l'arrivée d'un nouvel état, provoqué par d'autres serveurs. Il retourne en mode RESTRUCT pour un nouveau cycle RESTRUCT-GRADIENT-ALEA-RESTRUCT (R-G-A-R);
- N3: dans le mode GRADIENT il ne s'agit plus de trouver une direction cible, mais de calculer, à chaque top, où se trouve la prochaine position du serveur courant, en suivant le gradient du CG local à sa région, et d'y aller (méthode *Serveur::calculCible_allerY*). Tant qu'aucun nouvel état n'apparaît (dû à des changements des autres serveurs), le serveur courant reste en mode GRADIENT. Il y a deux façons de sortir de ce mode suivant que les échecs dépassent ou non un seuil défini par la variable de classe *Serveur::S_d_echec*;
- N5/5b: le serveur en mode GRADIENT a atteint le minimum local de sa région avec des échecs persistants au-delà du seuil. Dans ce cas la solution trouvée n'est pas suffisamment bonne. Le serveur passe en mode ALEApo. L'idée est de faire jouer des déplacements aléatoires dans la région courante (dans la limite de l'enveloppe de recouvrement) afin d'explorer des états plus intéressants. Le serveur ne passe pas directement en mode ALEA. Ce mode intermédiaire (ALEApo) permet d'attendre la fin de la PO en cours (serveur courant) pour passer en mode ALEA;
- N8: pour la région du serveur courant, le minimum local a été atteint et il y a peu d'échecs (ils sont inférieurs au seuil *S_d_echec*), le serveur courant passe en mode EQUILIBRE;
- N6: une fois arrivé en mode ALEA, le serveur courant y reste en se déplaçant vers la cible tantqu'un nouvel état n'a pas été trouvé (c'est le but de ce mode) et qu'un seuil (EXPLO_CL_ETATS2) n'a pas été atteint par

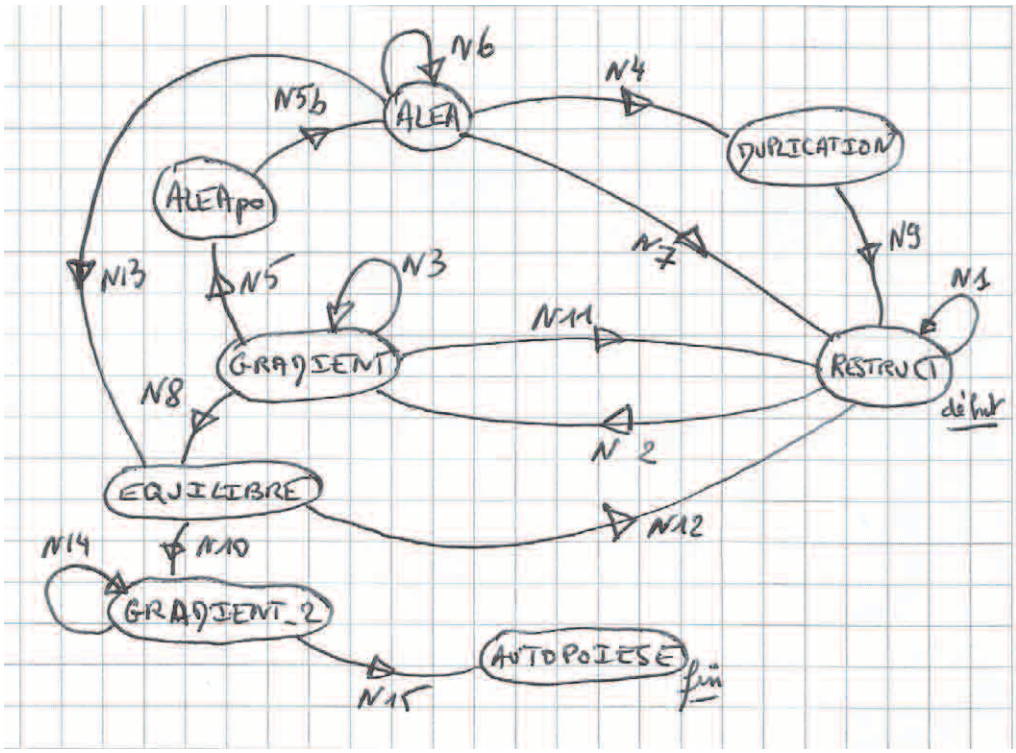


Fig-IV-E1-112

le compteur local au serveur (la variable de classe *cpt_explo_etats2*). C'est le même mécanisme qu'en mode RESTRUCT, pour exploiter suffisamment ce mode sans y rester piéger indéfiniment ;

- N4 : le serveur vient de dépasser le seuil EXPLO_CL_ETATS2 sans avoir provoqué de nouvel état. La méthode de déplacement aléatoire à l'intérieur de la région ne donne plus de résultat, il faut envisager une nouvelle technique. Le serveur passe en mode DUPLICATION, c.-à-d. qu'un nouveau serveur (en mode RESTRUCT) sera créé dans le système, entraînant un changement du problème ;
- N7 : dans le mode ALEA, bien que le serveur courant soit sous le seuil EXPLO_CL_ETATS2 et que de nombreux échecs persistent, si un nouvel état apparaît, alors le serveur courant ira directement en mode RESTRUCT ;
- N13 : le serveur courant est en mode ALEA. Bien que son compteur (*cpt_explo_etats2*) soit en-dessous du seuil, indiquant que l'exploration dans ce mode peut continuer, il passe en mode EQUILIBRE, car le nombre d'échecs devient inférieur au seuil (*S_d_echec*) ;
- N9 : une fois la duplication du serveur courant effectuée, ce dernier repart en mode RESTRUCT, après avoir fait une fois le cycle R-G-A-R ;
- N10 : la solution à l'équilibre sera potentiellement améliorée en lançant une nouvelle descente de gradient dans le mode GRADIENT_2 ;
- N14 : la descente de gradient fonctionne avec la méthode *Serveur::calculCible_allerY* ;
- N15 : lorsque le minimum local a été atteint le serveur passe en mode AUTOPOIESE. Il a fini d'évoluer.

Pour mesurer l'impact de cette nouvelle version v5, intégrant un comportement asynchrone des serveurs, prenons un exemple avec sept clients et deux serveurs (C_7S_2).

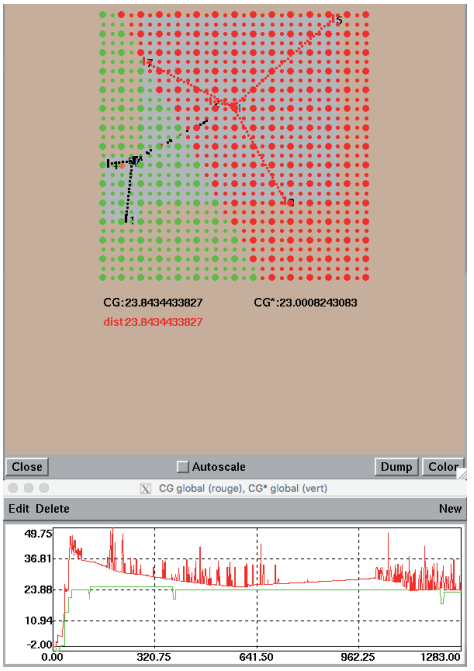


Fig-IV-E1-113

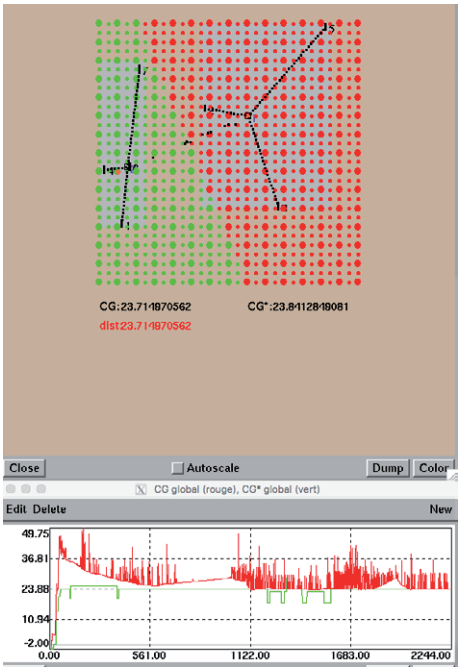


Fig-IV-E1-114

Les coordonnées des clients et (au départ) des serveurs sont les suivantes :

```
Client1(1.20052319728,3.19337666183);
Client2(5.70235267826,9.44146359406);
Client3(9.657888742,4.13608686912);
Client4(0.239430409968,6.10690032603);
Client5(,12.2124013781,13.8299611787);
Client6(,1.62636270264,6.27794319125);
Client7(,2.1556237499,11.5683646079);
Serveur1(,7.13949547482,13.5004452493);
Serveur2(,3.98330427612,13.3949688046).
```

Nous connaissons la solution optimale, calculée en 'C', avec les outils présentés au chapitre III :

- $CG^*_{7,2}$=23.000824308… (=4.267…+18.7337…) ;
- il existe 353 220 $((((2*15)-1)^2!)/(2!(((2*15)-1)-2)!))^4$ bipoints MM_{2-7} (états) possibles ;
- le bipoints optimal est $MM_{2-7}{}^*=\{P_1,P_2\}=\{(1,6),(6,9.5)\}$ sur une grille de deux-cent-vingt-cinq (15x15) points avec une maille de ½.

Nous avons lancé la simulation avec la recherche automatique de la période d'observation (PO) des serveurs, en mode coopératif (échange de PO entre serveurs). Il a fallu augmenter la vitesse ('V') de déplacement des serveurs, de 1/500 à 1/100, pour obtenir des résultats intéressants. Nous avons arrêté la simulation au top 1284. Elle a traversé trois états dont l'état courant, qui correspond à l'état optimal, cf. la figure *Fig-IV-E1-113*, $CG^*_{7,2}$=23.008243083. On voit également sur cette figure que la valeur trouvée de $CG_{7,2}$ n'est pas optimale, puisqu'elle vaut $CG_{7,2}$=23.8434433827. On peut dire que la résolution est passée, par «hasard», par l'état optimal. Le problème est qu'elle ne s'est pas stabilisée sur un état. Le serveur de couleur rouge est revenu au mode RESTRUCT, c.-à-d. qu'il utilise l'algorithme de contraction/expansion pour trouver de nouveaux états, après avoir été en mode GRADIENT, puis ALEA. Le serveur de couleur verte est passé du mode RESTRUCT (n°1) au mode

4 La formule (7) du chapitre III ($(D^2!/(K!(D^2-K)!))$) travaille sur une grille (D*D points) avec une maille de 1 par défaut. Pour une maille de ½, il faut remplacer 'D', dans cette formule, par (2D-1).

GRADIENT (n°2), puis du mode ALEA (n°4) au mode EQUILIBRE (n°6), pour terminer avec le mode AUTOPOIETIQUE (n°8).
Le mode RESTRUCT explique que la résolution n'est pas restée sur l'état optimum, bien qu'il ait été atteint. La preuve est apportée par la figure *Fig-IV-E1-114* sur laquelle, au top 2245, la même résolution est partie dans un nouvel état, par définition moins bon ($CG^*_{7,2}$=23.8412849081) que l'optimum trouvé précédemment.
Nous voyons sur la figure *Fig-IV-E1-115* les changements de mode des deux serveurs. Il aura fallu attendre une période longue (autour du top 1700) pour que le serveur de couleur rouge finisse par quitter le mode RESTRUCT pour rejoindre, par étapes, le mode AUTOPOIETIQUE. Cette expérience montre que la résolution n'est pas réflexive, dans le sens où elle ne s'arrête pas dans l'état optimum traversé, pour la simple raison qu'elle ne sait pas qu'il est optimum. Cela pose un problème au niveau du processus d'exploration des états du système. En effet, à quoi bon explorer ces états si la résolution ne peut pas reconnaître l'optimum absolu traversé, ou au moins l'optimum local. Avant d'en tirer les conséquences qui vont nous conduire à la v6 de la simulation, nous reportons, ci-après, les différentes expériences faites sur cet exemple :

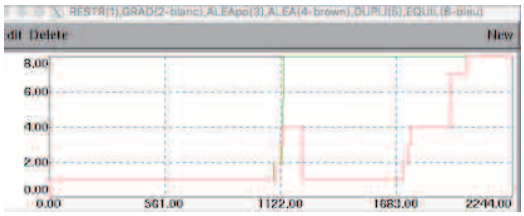


Fig-IV-E1-115

C_7S_2	top (~) stable	mode S1 (rouge)	mode S2 (vert)	Vitesse srv (V)	nbr d'états traversés	$CG^*_{7,2}$	$CG_{7,2}$
exp.1	1500	EQUILIBRE	EQUILIBRE	1/500	3	**23.0008**	**22.872**
exp.2	2500	EQUILIBRE	EQUILIBRE	1/500	8	23.841	23.719
exp.3	3000	EQUILIBRE	EQUILIBRE	1/500	6	23.841	23.715
exp.4	580	EQUILIBRE	EQUILIBRE	1/500	2	25.161	39.75
exp.5	900	EQUILIBRE	EQUILIBRE	1/200	2	25.161	25.84
exp.6	2320	AUTOPOIETIQUE	AUTOPOIETIQUE	1/200	3	23.726	23.352
exp.7	1284	RESTRUCT	AUTOPOIETIQUE	1/200	2	**23.0008**	23.843
exp.7	2200	AUTOPOIETIQUE	AUTOPOIETIQUE	1/200	3	23.841	23.714

Nous avons fait varier la vitesse de déplacement des serveurs. La simulation fonctionne et conduit la résolution à se stabiliser sans s'arrêter là où nous l'aurions souhaité. Nous sommes sur le bon chemin, car l'introduction de l'asynchronisme comportementale n'a pas rendu inopérante la résolution. Il nous reste cependant à affiner cette proposition.
Ces expériences nous poussent donc à faire évoluer la simulation pour lui donner cette capacité de réflexivité qui lui permettrait de s'arrêter lorsque ce serait le bon moment et de se caler sur le meilleur état qu'elle aurait traversé. Par opposition à l'approche calculatoire, évoquée au chapitre III, l'approche «naturelle» (exploratoire) que nous proposons ici ne conduit pas à trouver nécessairement l'optimum absolu (la meilleure de toutes les solutions possibles, au sens mathématique), mais l'optimum rencontré, c.-à-d. la meilleure des solutions rencontrées. Cela n'interdit pas de croiser l'optimum absolu, comme ce fût le cas avec l'exemple C_7S_2, présenté juste avant.
La différence est fondamentale puisqu'il s'agit d'intégrer dans la résolution des contraintes (topologiques ou autres) du problème. C'est dans ce sens que l'on parle d'une approche «naturelle», à l'image des phénomènes physico-chimiques observés sur terre.

IV-E2) La simulation version v6 : MVB[1]

Comme nous l'avons abordé précédemment, cette nouvelle version v6 (MVB) a pour objectif principal de rendre la simulation «réflexive» et surtout autopoïétique. Plusieurs modifications vont être nécessaires, en tout premier lieu le moteur comportemental (graphe des états internes) des serveurs.

Déplacement constant versus proportionnel

Avant d'aborder cette question du moteur comportemental, revenons sur un élément déterminant qui concerne le déplacement des entités (serveurs et requêtes) dans MVB. Des choix précis ont été faits. Ils conditionnent le bon fonctionnement des résolutions, où la vitesse de déplacement est un socle sur lequel se construit la cognition. Nous reviendrons ultérieurement sur cette notion fondamentale qui nous est très chère.
Or la vitesse de déplacement peut être gérée par une valeur constante (INV_PAS_DEPL_SRV1), ou comme nous allons le voir, dans cette nouvelle v6, possiblement de façon variable, voire obtenue dynamiquement par coopération entre les serveurs.

Au-delà de l'idée de moduler ou non la vitesse, nous avons travaillé plus finement sur la notion de déplacement des serveurs et des requêtes, de telle sorte qu'ils soient constants ou variables, indépendamment de la vitesse retenue au départ, et en tenant compte des phases de la résolution. Bien entendu, un déplacement est considéré comme constant pour une vitesse donnée fixe. À l'inverse, il peut être variable, même avec une vitesse fixe.

La figure *Fig-IV-E2-116* illustre le déplacement constant des requêtes (R_1 et R_2) vers le serveur cible (S_2), et ce, quelque soit la distance des requêtes à la cible.
Nous avons les données suivantes, issues de la simulation :

- $dx_1=x_{cible}-x_1$; $dy_1=y_{cible}-y_1$;
- $dxCorrige_1=dx_1/deplacement$, avec
- deplacement=$distance\_cour_1$/step, et
- step=TAILLE_REQUETE/INV_PAS_DEPL_REQ.

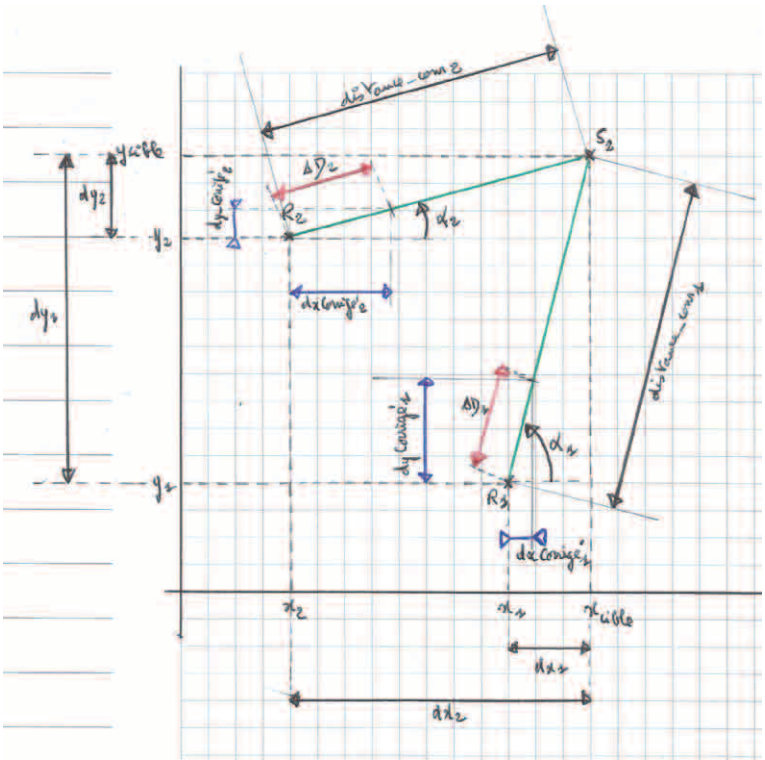


Fig-IV-E2-116

Or dans la simulation TAILLE_REQUETE vaut 1 et INV_PAS_DEPL_REQ vaut 5. Ces valeurs peuvent être positionnées autrement.
On en déduit que la variable *step* vaut 1/5 et que la variable *deplacement* vaut 5*$distance\_cour_1$.

1 Nous donnons le nom MVB (Maturana-Varela-Bourdon) à notre simulation en hommage aux biologistes Humberto R. Maturana et Francisco J. Varela pour leurs travaux sur l'autopoïèse, dont on trouvera des fondements dans «L'Arbre de la connaissance. Racines biologiques de la compréhension humaine.» aux éditions Addison-Wesley, 1992.

Nous avons, sur la figure *Fig-IV-E2-116* :

$\cos\alpha_1 = dx_{Corrige1}/\Delta D_1 = dx_1/distance\_cour_1$, or

$dx_{Corrige1} = dx_1/(5*distance\_cour_1) \Rightarrow$

$(dx_1/5*distance\_cour_1)*(1/\Delta D_1) = dx_1/distance\_cour_1 \Rightarrow$

$\Delta D_1 = 1/5 = 0.20$, $\forall \alpha_i$.

Cette technique permet d'assumer un déplacement constant des entités quelque soit l'angle (α_i), qu'elles font avec leur cible.
Dans la simulation, les requêtes se déplacent vers les serveurs, et les serveurs se déplacent, eux, soit vers d'autres serveurs, soit vers des clients.

Nous avons choisi un déplacement constant dans presque toutes les situations, sauf une:

- les requêtes se déplacent vers leur cible en utilisant la méthode *Requete::aller_vers* qui utilise un déplacement constant (INV_PAS_DEPL_REQ), comme le montre la figure *Fig-IV-E2-116*. Cela permet d'assumer un comportement cohérent entre toutes les requêtes de la simulation et ne pas introduire de biais dans le processus d'ajustement des serveurs;
- dans le cas des serveurs, nous avons plusieurs configurations pour leurs déplacements:
  1. en expansion dans le mode RESTRUCT;
  2. en contraction, toujours dans le mode RESTRUCT;
  3. dans le mode ALEA;
  4. dans le mode GRADIENT/GRADIENT_2.

Quand les serveurs sont en expansion dans le mode RESTRUCT (cas 1.), ou dans le mode ALEA (cas 3.), l'objectif est d'explorer l'espace des états pour découvrir de nouveaux états. Dans tous ces cas il est intéressant de recourir au déplacement constant. Pour cela on utilise la constante INV_PAS_DEPL_SRV1 pour l'expansion, la constante INV_PAS_DEPL_SRV2 pour l'ALEA et la méthode *Serveur::aller_vers* dans les deux cas (1. et 3.). Les simulations présentées, avec la v6, utilisent la même valeur de 1/40 pour ces deux constantes. Le fait de disposer de plusieurs constantes permet de faire une étude sur l'impact d'un déplacement constant mais différent suivant l'expansion en mode RESTRUCT (cas 1.) et le mode ALEA (cas 3.).

Dans les modes GRADIENT/GRADIENT_2 (cas4.), l'objectif pour les serveurs est de descendre vers l'optimum local de la région courante. La méthode utilisée dans ce cas n'est pas *Serveur::aller_vers*, mais *Serveur::calculCible_allerY,* qui travaille à déplacement constant, mais pas avec la même technique que dans les cas précédents. En effet ici on part d'une position courante (xcour/ycour) et on teste, dans l'ordre, les quatre positions voisines:

$PV_i = \{(xcour+0.1, ycour), (xcour, ycour+0.1),$
$(xcour-0.1, ycour), (xcour, ycour-0.1)\}$

Là encore le déplacement constant donne une régularité dans la descente vers l'optimum local. On pourrait jouer sur cette valeur pour aller plus ou moins vite vers le but.

Le dernier cas étudié (2.) concerne les serveurs en contraction dans le mode RESTRUCT. Le principe consiste à

rapprocher un serveur de l'un de ses clients avec lequel il a la plus forte demande. C'est le seul cas dont le déplacement ne doit pas être constant. En effet, plus le serveur approche du client et plus son déplacement doit décroître. Cela permet de prolonger cette phase sans avoir à gérer l'arrivée pile sur la cible.
Pour obtenir ce résultat nous utilisons la constante INV_PAS_DEPL_SRV1, comme pour l'expansion, mais sans utiliser, dans le dénominateur de la formule du calcul de la cible, la distance courante à la cible (distance_cour$_i$) :

- x_{cible}=x-(dx/INV_PAS_DEPL_SRV1); // déplacement proportionnel
  au lieu de:
- x_{cible}=x-(dx/(INV_PAS_DEPL_SRV1*distance_cour)); // déplacement constant

Protocole de coopération entre les serveurs

Pour introduire dans MVB un mécanisme autopoïétique il faut rendre la simulation «réflexive». Nous avons commencé à parler des serveurs comme des oiseaux dans une nuée (on parle de murmure ou murmuration). Une nuée impose des règles communes à ses membres ("boids"[2]), tout en autorisant des comportements propres non contradictoires avec les règles communes. En quelque sorte ils s'additionnent.

Les règles communes, transposées dans MVB, se cristallisent autour du graphe des états internes des serveurs (cf. la figure *Fig-IV-E2-118*). Chaque serveur partage ce graphe, c'est le patrimoine génétique commun. Rien ne lui interdit de l'utiliser de façon singulière (à la fois synchrone et asynchrone[3]) avec les autres serveurs. Nous verrons que la vitesse, en particulier, et le déplacement, en général, permettent l'émergence de ces façons singulières d'interprétation du patrimoine commun.

Dans une nuée «naturelle», on peut observer deux catégories. La première que j'appellerai simplement une «nuée ouverte» et la seconde une «nuée fermée».
Dans les deux cas les nuées possèdent des caractéristiques émergentes et globales, qui dépassent chaque individu y appartenant («le tout dépassant la somme des parties»). C'est, par exemple, le cas de la direction de la nuée, de sa densité[4] ou encore de sa vitesse de déplacement[5].

En revanche, ce qui les différencie c'est le fait qu'une «nuée ouverte» soit complètement poreuse à son environnement alors qu'une «nuée fermée» ne le soit pas. Par porosité j'entends le fait que des individus de la nuée, dans des proportions relativement faibles, peuvent aller et venir dans et en dehors de la nuée, sans changer la nature de

---

2 Craig W. Reynolds a développé en 1986 des entités informatiques, appelées "boids", simulant le vol des nuées d'oiseaux. Pour cela il a établi trois règles communes à toutes ses entités (non éloignement, non rapprochement et une direction commune).

3 clin d'œil à la mécanique quantique…

4 Un oiseau, seul, n'a rien à voir avec la densité d'une nuée, puisque cette dernière dépend du vide laissé entre les oiseaux dans la nuée. C'est assez semblable avec la densité d'une molécule vis-à-vis des électrons, des neutrons et autres composants élémentaires la composant. En éliminant le vide entre ces entités élémentaires on obtient un trou noir où la matière s'effondre sur elle-même.

5 Au temps 't' la vitesse d'un oiseau peut être opposée en direction et en valeur à celle de la nuée à laquelle il appartient.

la nuée. Ce mécanisme est différent et non contradictoire avec la possibilité, pour une «nuée ouverte», de se transformer, par exemple, en plusieurs nuées.
Une «nuée fermée» fonctionne avec un nombre d'individus qui ne va pas évoluer, ou peu, pendant son existence. Un exemple de «nuée ouverte» est celle des étourneaux, alors qu'un exemple de «nuée fermée» est celui du vol des oiseaux en 'V'.
Cette métaphore me semble pertinente dans le cadre de MVB dans la mesure où nous cherchons à mettre en place un dispositif réparti qui permet à la simulation d'explorer le graphe des états du système, tout en mémorisant l'état le plus intéressant traversé.
Rappelons qu'un état, ou solution, du système d'exploration correspond au positionnement de 'j' serveurs parmi 'i' clients (C_iS_j), et qu'à cet état nous pouvons associer/calculer une valeur CG appelée le coût global de l'état.
Un état est dit «intéressant» s'il se rapproche de l'optimum global (CG^*), c.-à-d. la meilleure solution, qui correspond, ici, à un coût minimal. Cela pourrait être autre chose. Or cet état ne peut être calculé qu'en faisant la somme des coûts locaux (CG_i) à chaque serveur ($région_i$). L'état est donc une grandeur globale qui définit, entre autres choses, la nuée des serveurs vis-à-vis de la résolution en cours.

état utile

Nous parlerons également d'état «utile», lors de l'exploration. Ce sont des états qui améliorent la valeur de la meilleure solution rencontrée pendant cette dernière résolution.
L'exploration consiste donc à emmener le système d'état en état jusqu'à avoir trouvé une solution satisfaisante.

Techniquement cela nécessite de recourir à un protocole qui permet aux serveurs de la nuée d'échanger «en permanence», entre eux, la valeur du coût local associé (CG_i) à leur région. Chacun peut reconstruire, par sommation, la valeur globale de l'état dans lequel le système se trouve.

donnée partagée

Cette valeur globale (CG) est traitée comme une donnée «partagée»[6] par les serveurs de la nuée. Pour maintenir la cohérence de ces données, il suffit, à l'un des serveurs de la nuée, de récupérer, auprès des autres serveurs, leur valeur locale, d'en faire la somme et de renvoyer cette somme à tous les serveurs.
Depuis la v3, nous récupérions la valeur des états traversés par le système, sans difficulté, grâce à la classe *Manager* qui centralisait, en temps réel, les résultats de la simulation pour en faire l'affichage sur des graphiques[7].
Maintenant il s'agit de faire ce calcul de façon décentralisée, et cohérente. C'est le rôle du protocole

6 Ce sont des données répliquées chez tous les serveurs de la nuée, dont le contenu unique est co-construit de façon cohérente. Contrairement à la mémoire partagée (segment de mémoire *IPC*) sur *UNIX/Linux*, qui est unique et accessible simultanément par différents processus (entraînant d'ailleurs des problèmes potentiels de cohérence des données lors d'accès concurrents), ces données sont multiples en mémoire (autant d'occurrences que de serveurs dans la nuée), mais avec un contenu identique. Cette règle impose que toute modification de l'une des occurrences est immédiatement répercutée aux autres occurrences. C'est ce que nous faisons dans le protocole de coopération. À l'image des codes correcteurs d'erreur, cette redondance est synonyme de surcoût, mais aussi et surtout, de grande fiabilité. Tant qu'il existe des éléments dans la nuée, la valeur est conservée.

7 Nous verrons sur un exemple précis comment l'affichage sur les courbes donne une idée précise, mais pas toujours exacte de ce qui se passe dans la simulation. En effet, le fonctionnement en mode multi-agents peut conduire, dans certains cas, à des pertes d'information au niveau de l'affichage, mais absolument pas au niveau des données internes de la simulation.

de coopération (méthode *Serveur::prot_coopCG*) exactement au moment d'un changement d'état par le système.
Plus précisément au moment où un nouvel état vient d'être détecté, mais pas encore mis en place.

Ce moment arrive par un seul serveur à la fois, lorsqu'une requête arrive en direct d'un client (on exclut ici les requêtes résiduelles en provenance d'un autre serveur) sur ce serveur et que ce dernier est capable (accepte) de traiter sa demande.
Cela provoque une cascade d'appels de méthodes (*Requete::main -> Serveur::lancer_calcul -> Serveur::maj_interaction -> Serveur::prot_coopCG*).
Le serveur concerné regarde (*Serveur::maj_interaction*) si le client, pour qui la requête est mandatée, est nouveau dans la liste interne de ses clients. Si c'est le cas il ajoute ce nouveau client à sa liste. C'est à ce moment précis que le système vient de changer d'état.

Ce changement est réel, y compris pour les serveurs qui ne seraient pas impactés directement. En fait, quand un nouveau client arrive sur un serveur via des requêtes, soit c'est un nouveau client pour le système (on trouve cette situation dans la phase d'initialisation de l'exploration quand les clients n'ont pas encore eu les premiers retours de leurs requêtes), soit c'est un client qui était avec un autre serveur avant cette interaction.
Cette deuxième situation est la plus fréquente, dans la mesure où l'autre cas ne concerne que les phases d'initialisation de l'exploration.
Dès qu'un client change de serveur, ce qui est dû à la mobilité des serveurs, au moins deux serveurs sont impactés, celui qui reçoit le client et celui qui le perd.

La méthode *Serveur::maj_interaction* gére ce changement dans tous les serveurs impactés. Avant de mettre à jour, et donc d'officialiser ce nouvel état (nouveau découpage en régions), cette méthode lance le protocole de coopération entre les serveurs pour calculer la valeur globale de l'état qui vient d'être formellement quitté.

Les serveurs de la nuée disposent, sous la forme d'une donnée «partagée», de la valeur (CGetat) du meilleur état jamais traversé par le système. En calculant l'optimum local de la région avant changement d'état, chaque serveur co-construit avec les autres la valeur globale de l'état qui vient d'être quitté. Il suffit, au serveur concerné par le déclenchement du changement d'état, de comparer la valeur obtenue avec CGetat, pour savoir si cet état est un état «utile» (meilleur que le meilleur trouvé jusqu'alors). Dans ce cas il échange la nouvelle valeur de CGetat avec les autres membres de la nuée.
Chaque serveur sauvegarde localement sa dernière position dans l'état qui vient d'être quitté, ainsi que la liste des clients constituant la région en cours. Ceci lui permettra dans le futur de rejoindre directement l'état traversé qui aura la meilleure valeur CG.

La dernière manipulation consiste à mettre à jour la liste «partagée» des états traversés (float[]Etats).

Le protocole de coopération, entre les serveurs de la nuée, nécessite que chaque serveur connaisse les autres serveurs de la nuée. Nous avons deux possibilités pour mettre en place ce protocole :

1. utiliser la méthode *(Serveur[])getAll("Serveur")* qui renvoie la liste de tous les serveurs de la simulation, impliquant qu'ils appartiennent tous à une unique nuée ;
2. utiliser les requêtes résiduelles entre les serveurs pour co-construire dynamiquement la liste des serveurs de la nuée (percolation).

Dans un premier temps, nous avons utilisé, dans MVB, la première technique, pour nous concentrer sur le protocole de coopération. Finalement nous avons introduit très rapidement la seconde technique (percolation), beaucoup plus pertinente, puisqu'elle peut tenir compte de la réelle évolution (coupure d'une nuée en plusieurs,…) de la nuée et nous affranchit de l'aspect centralisé disponible dans la simulation, qui est en contradiction avec nos objectifs de résolution distribuée.

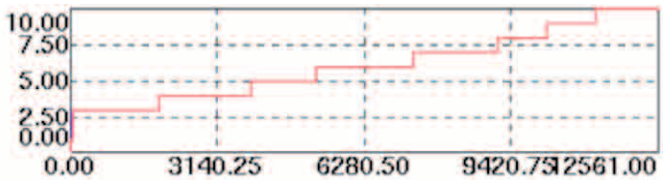


Fig-IV-E2-116b

La figure *Fig-IV-E2-116b* montre en abscisse les tops logiques de la simulation, et en ordonnées deux courbes qui se superposent. En bleu le nombre de serveurs (dix au total) qui sont créés dans la simulation et en rouge le nombre de serveurs reconnus (par percolation) comme faisant partie de la nuée.
Les deux courbes se confondent assez vite dès le départ, montrant ainsi l'intégration très rapide des nouveaux arrivants dans la nuée. En réalité il faut très peu de tops, à la simulation, pour co-construire la liste complète des serveurs de la nuée. En effet, la simulation commence, pour tous les serveurs, par la phase d'exploration (mode RESTRUCT, algorithme contraction/expansion). C'est l'occasion, pour les clients, de générer de nombreuses requêtes résiduelles (celles qui ont échoué avec leur serveur «au plus près»), à cause des serveurs qui, pour la plupart, en sont encore à mettre au point (processus d'ajustement différé) leur production sur la demande qui leur ait faite. La percolation joue à plein pendant cette phase d'initialisation. Elle ne coûtera plus rien par la suite, sauf au départ et à l'arrivée de serveurs dans la nuée.

Co-construction de la liste des membres de la nuée

Le principe adopté ici, pour cette co-construction (percolation) de la liste des membres de la nuée est le suivant :

- chaque serveur possède la liste «partagée» des éléments de la nuée (ses confrères). Au début de la résolution cette liste est vide pour tout le monde ;
- à chaque appel au protocole de coopération (méthode *Serveur::prot_coopCG*) sur un membre de la nuée, déclenché via la méthode *Serveur::maj_interaction* lors de l'arrivée sur ce membre d'une requête non résiduelle

(en provenance directe d'un client), le serveur concerné met à jour sa liste locale de membres de la nuée. Cette mise à jour est incrémentale, c.-à-d. que seuls de nouveaux membres seront ajoutés. Le cas où des membres quitteraient la résolution en cours (par exemple une panne d'un serveur) n'est pas pris en compte dans ce protocole, mais pourrait l'être sans difficulté. Il suffirait qu'avant de quitter la nuée, le serveur se retire de la liste «partagée» des membres de la nuées et échange cette nouvelle liste avec les autres. Pour ajouter d'éventuels nouveaux membres, le serveur courant regarde les serveurs qu'il voit à n+1, via les requêtes résiduelles qu'il reçoit, et pour chacun d'eux ceux qui sont vus à n+2. On aurait pu rendre ce principe récursif et aller au-delà de la profondeur n+2, mais cela n'est pas nécessaire, car tous les serveurs en font de même, dès qu'ils déclenchent le protocole de coopération. En très peu de temps le serveur courant voit si de nouveaux serveurs sont apparus par rapport à la liste «partagée» des éléments de la nuée. Dans ce cas il ajoute ces éléments à la liste et la met à jour chez tous les serveurs de la nuée. Ce surcoût est marginal, puisque dès que la liste est complète, plus aucune mise à jour ne sera nécessaire. Il ne peut y avoir d'incohérence dans la mise à jour de cette liste «partagée», puisqu'un changement d'état ne peut être détecté que par un serveur à la fois. De plus la mise à jour étant incrémentale, le seul risque encouru serait de vouloir ajouter plusieurs fois le même serveur à la liste; ce qui mènerait au même résultat.

En revanche cette méthode, par percolation, est un peu plus lourde à implanter, bien que le surcoût en fonctionnement, comme nous l'avons évoqué, reste faible. Par contre elle est indispensable dans le cas de grands systèmes C_iS_j à résoudre, où une vision centralisée n'est plus possible, car plusieurs nuées (notion inconnue du simulateur dans la version actuelle des simulations) pourraient se former dynamiquement en fonction des résolutions.
Dans de telles situations, la période de co-construction des listes des membres des nuées pourrait introduire un temps de «flottement» plus grand, pendant lequel les résultats (calcul du CG) ne seraient plus pertinents.

Nous avons déjà rencontré ce phénomène avec les résolutions présentées au moment de leur initialisation, lorsque tous les clients ne sont pas encore connectés à un serveur. Les valeurs obtenues (CG) sont inférieures aux valeurs optimales; il ne faut donc pas en tenir compte, malgré que leur valeur (CG) soit meilleure (<) que celles qui seront trouvées quand tous les clients seront connectés aux serveurs.
Pour corriger ce problème, on regarde l'évolution du nombre de clients entre chaque nouvel état. Si le nombre augmente cela signifie que l'état précédemment calculé n'est pas pertinent et nous réinitialisons la valeur (CG) trouvée, même si sa valeur est supérieure. Cette technique fonctionne bien, mais elle nous fait perdre la valeur CG du premier état valide parcouru. Une astuce nous permet de conserver cette valeur et de l'afficher avant de dupliquer

un serveur pour faire une nouvelle résolution. Nous ne pouvons pas traiter ce premier état valide comme les autres. Cela n'est pas grave dans la mesure où il est très peu probable que le premier état valide parcouru soit l'optimum. Si cela était le cas, malgré tout, il serait encore moins probable que le système ne repasse pas par ce premier état optimal, lors de ses explorations à venir.

Pour revenir aux notions de «nuée ouverte» et de «nuée fermée», un point distinctif important consiste à optimiser une ou plusieurs grandeurs pour la seconde, alors que la première se contente d'un fonctionnement collectif adapté. Le vol en 'V' cherche à minimiser l'effort de chaque individu lors d'un vol au long cours.
Dans ce cas chaque oiseau connaît les autres oiseaux et les voit. Cela leur permet de pratiquer l'alternance des positions dans le 'V'. En effet un désordre quelconque prolongé pourrait remettre en cause l'efficacité de l'ensemble. Dans la nuée d'étourneaux, les oiseaux ne sont pas tous reliés entre eux et n'ont même pas l'idée de l'existence d'une grande partie des autres membres de la nuée. La porosité (entrées et sorties des oiseaux dans la nuée) ne perturbe en rien la nuée, pourvu qu'elle soit faite dans des proportions «raisonnables».

Il en est exactement de même avec MVB, puisque si l'on cherche à trouver la solution globale optimale (CG^*) on se rapproche d'une «nuée fermée» dans laquelle tous les serveurs vont se connaître pour optimiser cette valeur partagée.
Il est donc cohérent et justifié de mettre en place ce protocole de coopération entre les serveurs de la nuée vis-à-vis des objectifs fixés. Nous verrons d'ailleurs dans nos expérimentations que ce protocole fonctionne très bien.

Pour comprendre le rôle et la mise en œuvre du protocole de coopération entre les serveurs, il faut revenir sur la dynamique de la résolution.
Les clients émettent des requêtes à la recherche de serveurs capables de produire les ressources nécessaires pour satisfaire leur demande. Les requêtes ont des capteurs qui leur permettent, en temps réel, de voir le serveur le plus proche et de s'en rapprocher, même si ce dernier est lui-même mobile.
Nous allons nous concentrer sur le cas où une requête est satisfaite par le serveur visé (ressources disponibles suffisantes).

Voici les différentes étapes qui se produisent à l'arrivée d'une requête à proximité d'un serveur:

1. *Requete::main*: dans cette méthode, déclenchée à chaque top du fonctionnement des requêtes, la requête vise le serveur le plus proche (méthode *DomaineManager::view_agt*) et se rapproche de ce dernier (déplacement constant). Lorsqu'elle arrive à une certaine distance de ce dernier, elle tente une demande de calcul (*Serveur::lancer_calcul*). Si le calcul est possible (serveur disponible avec suffisamment de ressources), la requête transmet l'information à son client et disparaît (*Requete::delete*). Sinon elle cherche un autre serveur au plus proche;

2. *Serveur::lancer_calcul* : le serveur regarde si la demande de la requête est satisfaisable pour lui. Si c'est le cas il lance la méthode *Serveur::maj_interaction* pour mettre à jour ou créer un objet *Interaction*, interne à ses données, lui permettant la gestion du client concerné. La méthode *Serveur::maj_interaction* ne fait rien si la requête est résiduelle, c.-à-d. qu'elle ne vient pas en direct d'un client, mais qu'elle est déjà passée par un ou plusieurs serveurs qui n'ont pas pu la traiter;
3. *Serveur::maj_interaction* : cette méthode regarde si le client annoncé est nouveau pour le serveur courant qui la déclenche. Si ce n'est pas le cas (client déjà en interaction avec lui), il n'aura qu'à mettre à jour ses données internes. Si le client est nouveau pour lui, **cela signifie que le système vient juste de changer d'état**. Le serveur déclenche le protocole de coopération entre tous les serveurs via la méthode *Serveur::prot_coopCG*;
4. *Serveur::prot_coopCG* : le système vient juste de rentrer dans un nouvel état. Il faut calculer et sauvegarder, chez tous les serveurs de la nuée, des informations liées à l'état qui vient d'être quitté. On regarde, en particulier, si l'état qui vient d'être quitté n'est pas le meilleur rencontré jusqu'alors.

Sur la figure *Fig-IV-E2-117* nous voyons le système qui passe de l'état $Etat_i$ à l'état $Etat_j$. La différence vient du fait que le serveur S_1 s'est rapproché du client C_3. Ce faisant il est devenu le serveur le plus proche pour le client C_3.

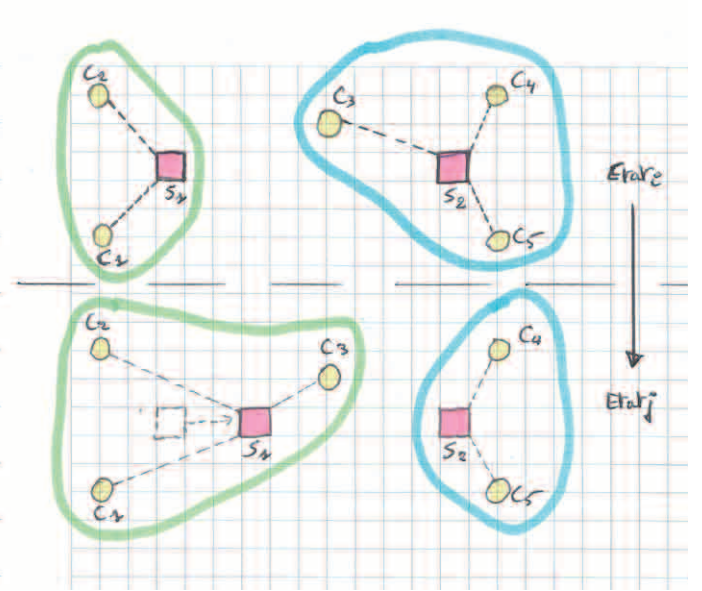


Fig-IV-E2-117

Le changement d'état du système ne peut être détecté qu'une fois constaté dans la méthode *Serveur::maj_interaction*. Dans l'absolu de la répartition des traitements informatiques par le simulateur *oRis*, plusieurs changements d'états pourraient arriver simultanément, occultant des changements intermédiaires. Malgrè le pseudo-parallèlisme de ces traitements, un seul changement d'état sera fait à la fois (mode *randomPicking* de la méthode *setPinckingMode* du moteur *oRis*).
Ceci nous garantit qu'aucun changement d'état ne sera oublié ou passé inaperçu. Nous reviendrons sur cette propriété fondamentale, lors de l'épilogue dans la partie IV-E3.
Dans la mesure où les conséquences du changement d'état (mise à jour des données internes des serveurs et leur répercussion chez les clients) sont traitées dans la méthode qui détecte le changement d'état, il suffit, avant les mises à jour et après la détection du changement d'état, de lancer le protocole de coopération pour calculer et partager des informations sur l'état que l'on vient de quitter.

L'objectif de l'exploration est de traverser les meilleurs états possibles (y compris l'optimum absolu CG*) en s'appuyant sur le graphe de comportement des serveurs, jusqu'à trouver la stabilité du système. Une fois atteinte, le système se cale sur le meilleur état rencontré pendant l'exploration. Cela signifie que chaque serveur a mémorisé sa dernière position (x/y) dans le meilleur état traversé et qu'il va revenir à cette position lors du calage. Tous les membres de la nuée vont ainsi aller simultanément[8] dans le meilleur état traversé depuis le début de l'exploration. Une dernière descente de gradient leur permettra d'atteindre l'optimum local dans chacune des régions constituant cet état, conférant au système la meilleure solution (CG) jamais traversée.
Le serveur, qui a détecté le changement d'état et que l'on nommera serveur courant, lance la méthode *Serveur::prot_coolCG* qui pilote le protocole de coopération entre les serveurs du nouvel état courant, afin de calculer et mémoriser localement à chacun d'eux la valeur globale CG de l'état qui vient d'être quitté, ainsi que les dernières coordonnées (xEtat_1/yEtat_1) du serveur dans cet état. Cette mémorisation n'aura lieu que si cette valeur CG calculée est meilleure (<) que la précédente, mémorisée par tous, de telle sorte que tous les serveurs ne conservent que le meilleur état traversé. Si l'état quitté est meilleur que tout ceux traversés jusqu'alors, chaque serveur mémorise en plus du CG correspondant, la liste de ses clients formant la région associée à cet état mémorisé.
Après avoir mis à jour la liste des serveurs de la nuée, la méthode *Serveur::calculeCgReg* est lancée auprès de tous, y compris sur le serveur courant lui-même. Cela met à jour, dans les données internes de chaque serveur, la valeur du CG local à sa région avant le changement d'état. Le serveur courant n'a plus qu'à récupérer l'ensemble de ces valeurs locales (CG de chaque région) pour calculer la valeur globale avant changement d'état. Si cette valeur est inférieure à la précédente mémorisée (la meilleure des anciennes explorations à n-2) il demande à tous les serveurs de mémoriser cette valeur ainsi que les clients de leur région associée.
Ce protocole permet, en fin de cycle d'exploration, de figer la résolution sur le meilleur état traversé (calage).

Modification du graphe des états internes des serveurs

Pour utiliser au mieux le protocole de coopération, corriger quelques défauts repérés dans la v5, et permettre à l'exploration d'aller en profondeur (passage de 'n' serveurs à n+1 avec 'n' compris entre 1 et le nombre de clients), nous avons fait évoluer le graphe des états internes des serveurs. La méthode *Serveur::restructurer* n'a subit aucun changement entre la v5 et la v6. Seule la méthode *Serveur::main*, qui code le graphe des états

8 Nous osons, ici, rapprocher ce changement collectif de comportement (aller vers un état partagé dont chaque membre à une place particulière) au phénomène d'intrication quantique, dans lequel des particules intriquées changent simultanément d'état (de façon corrélée), quelque soit la distance physique qui les sépare. On peut dire que les membres d'une nuées sont intriqués, dans la mesure où, simultanément, ils vont agir sur un état global partagé (leur position dans l'état global optimal rencontré).

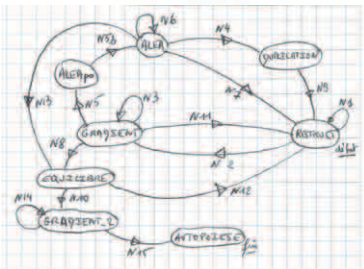


Fig-IV-E1-112 (rappel) v5

Fig-IV-E2-118

internes des serveurs, a été modifiée. On retrouve le graphe de la v5 (cf. la figure *Fig-IV-E1-112*) avec de très nombreux apports, comme le montre la figure *Fig-IV-E2-118*.

Le nouveau graphe possède seize états (de -5 à 10 en orange) et trente-quatre transitions (de N0 à N33 en jaune). Tous les états négatifs ont été ajoutés, pour mettre en place le processus autopoïétique de fermeture du système.
Une autre différence importante avec la v5 concerne l'état DUPLICATION. En effet cet état est atteint uniquement lorsque l'exploration a trouvé la meilleure solution pour les 'k' serveurs (membres de la nuée) du système. Tous ces 'k' serveurs se synchronisent sur leur état interne '-4' et vont tous passer en mode DUPLICATION (via la transition N26). Le premier, dans ce mode, provoque la création d'un k+1e serveur. L'exploration va pouvoir reprendre (N29) son cours avec un élément supplémentaire dans la nuée.

Avant de détailler les différents états et leurs transitions associées, de cette nouvelle mécanique, rappelons que l'objectif est de trouver toutes les configurations «optimales» aux différents niveaux de profondeur allant d'un élément (serveur) dans la nuée, à 'n' éléments, où

'n' est le nombre de serveurs du système, ne pouvant excéder celui du nombre de clients.
Pour cela les serveurs du système de résolution doivent se synchroniser, sans aucun contrôle centralisé, pour trouver ces 'n' différentes solutions. Nous allons nous concentrer sur les principaux apports de cette v6. Dans la suite du document nous emploierons indifféremment les termes «mode» ou «état interne» pour rendre compte du comportement interne des serveurs pendant la résolution.

Voici donc les différents états internes des serveurs:

- mode RESTRUCT: ce mode est la clé de l'exploration de l'espace des états du système. Il met en œuvre l'algorithme de contraction/expansion qui permet cette exploration de façon pertinente. Plusieurs écueils doivent être déjoués lors de cette exploration dans ce mode. En effet, il suffit qu'un seul serveur boucle dans ce mode pour que toute l'exploration soit bloquée. Ceci découle des mécanismes de synchronisation mis en place pour franchir certains états (à l'image de l'état 'Z' dans les sémaphores sur *Linux*). Un deuxième écueil consiste à ne pas sortir des optima locaux. Enfin le troisième écueil empêche la sortie d'une boucle, non plus à l'intérieur du mode RESTRUCT, mais entre plusieurs modes, comme par exemple entre les modes RESTRUCT et ALEA, via les modes GRADIENT et ALEApo. Quatre seuils ont été créés (EXPLO_CL_ETATS1, EXPLO_CL_ETATS1_1, EXPLO_CL_ETATS3 et EXPLO_CL_ETATS4) pour sortir de ces écueils. Le premier seuil mesure l'absence d'arrivée de requêtes résiduelles, qui traduit une exploration inefficace. En effet on cherche à trouver de nouveaux états «utiles» qui sont provoqués par la création de ces requêtes résiduelles. Un compteur est incrémenté à chaque nouvelle requête résiduelle acceptée par le serveur courant. Un autre compteur est incrémenté à chaque top logique, si le compteur précédent reste constant. Ce compteur est remis à 0 dès qu'une requête résiduelle arrive et est acceptée par le serveur concerné. Un troisième compteur est incrémenté si le serveur courant ne bouge pas entre deux tops logiques consécutifs. Ce compteur, grâce à son seuil associé, permet de sortir d'un état d'équilibre où les serveurs ne se déplacent plus, interdisant toute nouvelle exploration. Enfin un quatrième compteur est incrémenté si aucun nouvel état n'est trouvé. Ce compteur ne tient pas compte des changements d'états qui renvoient le système dans un état déjà traversé. Cela évite les oscillations entre des états connus. Tant qu'aucun des différents seuils n'est atteint et qu'aucun nouvel état «utile» ne soit trouvé, le serveur reste en mode RESTRUCT (N1), et l'exploration continue. Sinon le serveur regarde s'il a dépassé la durée maximale autorisée dans ce mode (quatrième seuil). Dans ce cas l'exploration ne donne plus rien, le serveur passe (N33) en mode EQUILIBRE. Pour tous les autres seuils dépassés, le serveur passe (N2) en mode GRADIENT. L'idée est que l'algorithme contraction/expansion ne donne plus rien, mais l'exploration peut être envisagée en introduisant de l'aléa avant de reprendre l'algorithme. Pour pouvoir tester l'utilité d'un état, chaque serveur conserve la liste

des états déjà rencontrés. Pour cela on mémorise la valeur du CG équivalent à chaque état, partant du principe que deux états «utiles» différents ont des valeurs CG différentes;

- mode ALEA: ce mode est utilisé pour sortir d'un minimum local, dans lequel le système est bloqué dans le mode RESTRUCT. En déplaçant le serveur concerné, dans une direction aléatoire à l'intérieur de sa région, et ce jusqu'à ce qu'un nouvel état «utile» arrive, le système sort de ce minimum local potentiel et repart (N7) en mode RESTRUCT. Deux seuils (EXPLO_CL_ETATS2, EXPLO_CL_ETATS2_1) fonctionnent comme pour le mode RESTRUCT pour éviter des bouclages dans le mode ALEA. Si les compteurs sont sous les seuils mais sans échecs (un échec est une requête qui n'a pas été satisfaite par le serveur), alors le serveur passe (N13) en mode EQUILIBRE. La fin provisoire de l'exploration est arrivée pour lui. En revanche il restera (N6) en mode ALEA s'il est sous les seuils, mais avec des échecs (au dessus d'un seuil). Enfin si aucun nouvel état n'a été trouvé et qu'au moins l'un des deux seuils a été atteint, le serveur doit tenter autre chose de plus radical. Il passe (N4) en mode RAZPOS qui provoque un changement aléatoire de sa position;
- mode EQUILIBRE: dès qu'un nouvel état «utile» arrive, alors que le serveur est dans le mode EQUILIBRE, ce dernier passe (N12) en mode RESTRUCT. Bien qu'il soit à l'équilibre, ce nouvel état signifie que l'exploration vaut le coup d'être reprise. Dans le cas contraire (pas de nouvel état) le serveur passe en mode -5;
- mode -5: ce mode permet de synchroniser tous les serveurs sur le dernier serveur qui va passer par le mode EQUILIBRE, puis -5. Là encore si l'un des autres serveurs provoque un nouvel état «utile», le serveur courant passe (N18) en mode RESTRUCT pour reprendre une exploration. Avant de passer (N19) en mode GRADIENT2, il faut vérifier si le premier et seul état, ou le dernier état traversé (si pas le seul), ne serait pas meilleur que les autres. En effet on fait ce test en changeant d'état pour celui que l'on quitte, mais on ne le fait jamais pour le dernier état rencontré ou le premier s'il est seul. Seul le premier serveur à passer dans ce mode va faire ce test avant les descentes de gradient individuelles (mode GRADIENT2);
- mode AUTOPOIESE et mode -1: dès qu'un serveur arrive dans le mode AUTOPOIESE il incrémente la variable de classe *nbSrvEnAtt* chez tous les serveurs de la nuée et passe (N20) en mode -1. Cette variable propre à chaque serveur, mais dont le contenu est «partagé», par échange de valeur dès que l'une d'entre elles est modifiée par un serveur, représente le nombre d'éléments de la nuée qui sont dans le mode -1, c.-à-d. prêts à lancer la fermeture de l'exploration. En effet l'autopoïèse/fermeture-du-système ne peut s'effectuer que si tous ses éléments ont fini leur cycle. À chaque top logique, les serveurs prêts regardent cette variable et attendent que sa valeur soit égale au nombre d'éléments de la nuée. Ils connaissent ce nombre via la liste des membres de la nuée qui est mise à jour à chaque changement. Dès que le dernier serveur de la nuée se trouve en mode -1, tous les autres, y compris ce dernier, passent (N22) en mode -2;

- mode -2: la phase finale de fermeture du système peut commencer. Chaque serveur se positionne dans le meilleur état traversé pendant l'exploration en utilisant les variables *xCGetat/yCGetat* correspondantes aux dernières coordonnées du serveur dans ce meilleur état traversé. Il va aussi mettre à jour sa liste des clients (région) dans la variable *cls_dom*, puis recalculer l'enveloppe de recouvrement (x/ydom,X/Ydom) correspondant à cette région. Il va aussi recalculer (méthode *Serveur::calculeCgReg*) la valeur du CG local à sa nouvelle région, en fonction de sa nouvelle position, qui n'est pas forcément celle de l'optimum local dans la région optimale. En quelque sorte, il va rendre ses données internes cohérentes vis-à-vis de ce changement brutal de position dans le système, afin de terminer le processus par une dernière descente de gradient. Il passe (N23) ensuite en mode -3;
- mode -3: le serveur lance une ultime descente de gradient (méthode *Serveur::calculCible_allerY*). Si le minimum local est atteint (retour de la méthode précédente) il passe (N24) en mode -4, sinon il reste (N10) en mode -3;
- mode -4: le processus de fermeture est terminé pour ce serveur, il termine son cycle d'exploration. Il se met en attente (N30) que tous les autres serveurs arrivent dans ce mode (nouveau point de synchronisation des éléments de la nuée). Dès que tous les serveurs atteignent ce stade, la résolution est terminée et le système est positionné sur la meilleure solution traversée pendant l'exploration; deux solutions sont alors possibles, soit repartir dans un cycle d'exploration en gardant le même nombre de serveurs (N27), pour tenter de trouver un meilleur optimum global, soit arrêter ces cycles et choisir une exploration en profondeur (k+1, où 'k' est le nombre actuel de serveurs). C'est le dernier serveur, qui atteint le mode -4, qui choisit la solution en fonction de la constante globale (NB_EXPLORATIONS) positionnée au lancement de la simulation. À chaque exploration un compteur «partagé» est incrémenté. Tant que le nombre d'explorations n'atteint pas la constante globale, les serveurs passeront (N27) en mode RESTRUCT. Quand le nombre d'explorations est atteint, les serveurs passent (N26) en mode DUPLICATION, pour faire des explorations au niveau k+1;
- mode DUPLICATION: c'est toujours le dernier serveur qui est passé en mode -4 qui déclenche la duplication d'un serveur. En réalité il crée un nouveau serveur. Se pose alors la question de l'endroit où créer ce nouveau serveur dans la nuée. L'idée n'est pas nécessairement de choisir la proximité immédiate du serveur qui crée ce nouveau serveur, mais plutôt de choisir la proximité du serveur de la nuée qui possède le plus de clients. En effet, plus on descend en profondeur, plus le nombre de serveurs se rapproche du nombre de clients, il ne faudrait pas que le système crée un nouveau serveur qui n'aurait pas de client. Cette création serait inutile. C'est la raison pour laquelle on prendra toujours (d'autres stratégies peuvent être imaginées) le serveur qui a le plus de clients. Une fois le serveur créé, il faut mettre à jour tous les afficheurs en ajoutant une courbe pour ce dernier.

Avant de regarder le fonctionnement de MVB, à partir d'exemples, et ouvrir les discussions vers de nouveaux horizons, nous allons rapidement faire un bilan chiffré et quantitatif de l'évolution des versions de la simulation depuis la v3 jusqu'à la v6 (MVB).

modules/versions (*oRis*)	**v3** **ligne**/mots/ caractères			**v4** **ligne**/mots/caractères			**v5** **ligne**/mots/caractères			**v6 (MVB)** **ligne**/mots/caractères		
Manager.ors	**562**	1997	20534	**610**	2202	22613	**610**	2202	22603	**629**	2262	23246
DomaineManager.ors	**601**	2767	24822	**603**	2763	25027	**610**	2820	25399	**684**	2855	28581
Serveur.ors	**1527**	7605	67735	**1626**	9061	84008	**1959**	11549	107137	**3294**	20571	210638
Requete.ors	**214**	855	8573	**258**	1133	12417	**262**	1306	13654	**263**	1191	13311
Client.ors	**470**	1697	16833	**475**	1728	17139	**477**	1730	17197	**550**	1836	20510
fusion.cste	**83**	1057	6478	**88**	1174	7189	**98**	1324	8251	**112**	1559	9787
fusion.def_classes	**570**	3842	30171	**613**	4152	32373	**619**	4303	33318	**656**	4760	36638
TOTAL	**4027**	19820	175146	**4273**	22212	200766	**4635**	25234	227559	**6188**	35024	342711

Le tableau ci-dessus reprend les principaux «modules» de la simulation placés dans des fichiers «.ors». Pour chacune des versions nous avons trois colonnes, respectivement le nombre de lignes (en gras), le nombre de mots et le nombre de caractères de ces modules.
Nous avons 4027 lignes pour la v3 (2013-2014), 4273 pour la v4 (2022), 4635 pour la v5 (2022) et 6188 pour la v6 (MVB-2023).
Le nombre de caractères est passé de 175146 caractères pour la v3, à 200766 pour la v4, puis 227559 pour la v5 et 342711 pour la v6 (MVB).
L'évolution des lignes et celle des caractères intègrent, sans discernement, le code et les commentaires. Il y a donc eu une augmentation des deux entre la v3 et la v6, globalement de 53.66% pour les lignes et de 95.67% pour les caractères. Avec l'expérience j'ai augmenté sensiblement les commentaires, me facilitant la reprise du code et la rédaction de ce mémoire.
La classe *Serveur* représente à elle seule plus de 50% du code.

Le passage à la v5, puis à la v6 a été «relativement» douloureux au niveau de la mise au point du code. Changer le moteur de la simulation fut un très grand défi. Bien que ni l'augmentation du volume du code, ni la perte inéluctable de neurones n'arrangeaient les choses, j'ai cru ne pas réussir à surmonter la mise au point de ces apports fonctionnels conséquents dans le code.

Je n'ai pas pu m'empêcher de penser à l'un de mes tous premiers articles[9] que j'avais rédigé après une certaine expérience en programmation objet, où je mettais en avant l'impact du code nouveau sur un code existant.
En réalité, entre la v3 et la v6, j'ai fait une faute d'analyse sur le comportement à donner aux entités de la simulation. C'est sans doute l'une des choses les plus difficiles à trouver et à corriger, car les répercussions peuvent être inattendues et relativement éloignées des causes.

9 "An Experiment in Prototyping Using the Object Model as Structuring Agent". François Bourdon, TOOLS'90, poster session, Technology of Object-Oriented Languages and Systems, Paris, 1990.

En fait j'avais voulu rendre hommage à *Linux* en implantant un mécanisme qui différait dans le temps la prise en compte d'un changement d'état du système. Exactement ce qui se passe dans *Linux* avec la fermeture des segments de mémoire partagés (*IPC*). Je trouvais ça très élégant, car l'objet partagé était marqué à détruire, jusqu'à ce que le dernier processus qui l'utilise se termine lui-même.

Dans mon cas il s'agissait des objets *Interaction*, propres à chaque serveur, qui stockent les informations correspondant à leurs clients (un objet par client). Le problème venait du fait que lorsqu'un client se tournait vers un autre serveur, le serveur préalablement relié à ce client devait détruire l'objet *Interaction* correspondant. J'avais donc imaginé que cette destruction serait marquée (méthode *Serveur::AD_interaction*-v5) au moment du détournement du client, est rendue effective (destruction de l'objet *Interaction* correspondant par la méthode *Serveur::del_interactions*) en fin de période d'observation (PO) du serveur concerné.
En réalité cela rendait incohérent ma simulation entre le marquage et la destruction effective. Cet hommage c'est transformé en catastrophe qu'il m'a fallu résoudre non sans mal…

Nous allons tester MVB sur quelques exemples significatifs, à commencer par C_7S_2, que nous avons pris avec la v5 et qui traversait l'état optimal sans s'y arrêter. Ensuite nous prendrons le cas de $C_{10}S_3$, avec lequel nous exploiterons toutes les nouvelles possibilités de MVB.

Sept clients et deux serveurs (C_7S_2) sur MVB

Reprenons exactement le scénario, vu avec la v5, avec sept clients (aux mêmes positions), et deux serveurs, le tout sur une grille de deux-cent-vingt-cinq (15x15) points (D_GRILLE=15) avec une maille (MAILLE) de ½.
Pour rappel, la solution globale optimale calculée en 'C' est :

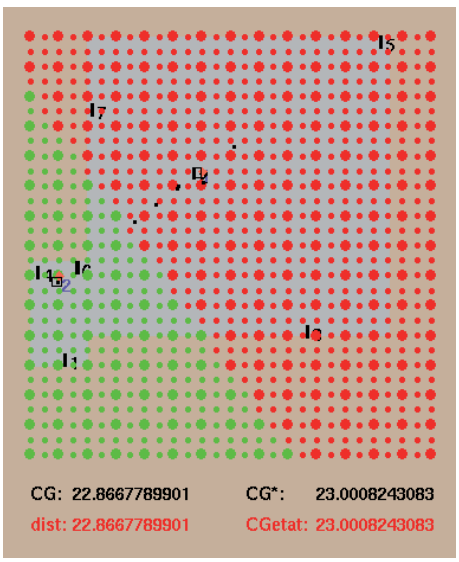


Fig-IV-E2-119

$CG^*_{7,2}$=23.000824308… (=4.267…+18.7337…), avec le bipoints optimal $MM_{2\text{-}7}{}^*=\{P_1,P_2\}=\{(1,6),(6,9.5)\}$.

Nous avons lancé MVB sur cet exemple en laissant l'exploration aller jusqu'à sa stabilité, l'atteinte du mode AUTOPOIESE pour tous les serveurs et la fin de la résolution. Nous pouvons déjà constater, sur la figure *Fig-IV-E2-119*, que le système est arrêté sur l'optimum global absolu (23.000824308…).
La valeur (CG) obtenue est même meilleure (22.8967789901…), puisqu'une fois l'état optimal atteint (mode -2), les serveurs font une ultime descente de gradient (mode -3), dont la précision dépend de la constante choisie dans l'algorithme (méthode *Serveur::calcule_allerY*), ici égale à «0.1*cosα».

Nous avons donc trouvé le bipoints
$MM_{2\text{-}7}{}^*=\{P_1,P_2\}=\{(0.909,5.803),(5.802,9.468)\}$, avec
$CG^*_{7,2}$=22.8667789901… (=4.22…+18.6429…).

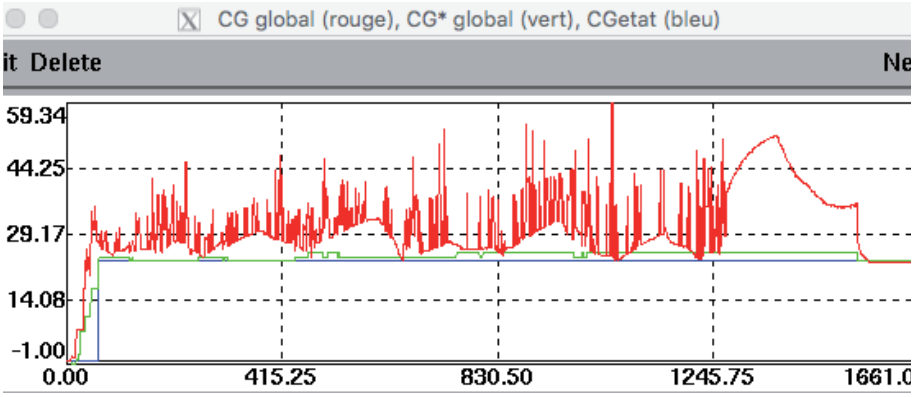


Fig-IV-E2-120

Sur la figure *Fig-IV-E2-120* on peut observer la fin de l'exploration, aux alentours du top logique 1250.
Le système se recale alors sur l'optimum traversé bien avant. Ici le système a exploré quatre états.

Pour bien comprendre le graphe de la figure *Fig-IV-E2-120* il faut préciser la signification des trois courbes :

- CGglobal (en rouge) : cette courbe correspond à la valeur de la solution globale (CG) au moment de l'exploration. Elle est calculée par le *Manager* en sommant les valeurs exactes (distance au serveur et débit) contenues dans chaque client de la simulation. Les pics vers le haut montre un système (position des serveurs vis-à-vis des clients) qui explore dans des directions non favorables ;
- CG*global (en vert) : cette courbe correspond à la solution optimale globale calculée par le *Manager* en sommant les valeurs obtenues par chaque serveur. Pour cela il s'appuye, non pas sur la position des serveurs au temps 't', mais sur celle des clients de leur région. Chaque serveur utilise la grille (et sa maille associée). Le calcul se fait sur l'enveloppe de recouvrement de la région. Cette courbe verte indique donc la valeur globale «optimale» de l'état en cours. C'est la raison pour laquelle la courbe rouge est le plus souvent au-dessus de la courbe verte ;
- CGetat (en bleu) : cette courbe bleue représente la valeur calculée du meilleur état rencontré au fur et à mesure de l'exploration.

Très vite l'état optimal a été traversé, comme le montre la courbe bleue (23.008…) de la figure *Fig-IV-E2-120*. Les descentes de gradient successives se repèrent avec les pics rouges vers le bas. La courbe verte se superpose avec la bleue au passage par l'état global optimal, pour s'en écarter par la suite.
Vers le top 1660, en fin de résolution (mode AUTOPOIESE), juste avant le calage du système sur la meilleure solution traversée, nous voyons nettement, sur la courbe verte (CG*global), que la meilleure solution de l'état en cours est moins bonne (valeur supérieure) que la meilleure rencontrée (courbe bleue).
Le moment du calage permet à la courbe rouge (CG effectif) de passer sous la bleue, grâce aux dernières descentes de gradient des serveurs.

Sur cet exemple le système fonctionne exactement comme nous le souhaitions. La constante INV_PAS_DEPL_SRV1, qui détermine la vitesse (V=1/INV_PAS_DEPL_SRV1) des serveurs vaut 40; ce qui fait une vitesse «relativement» rapide.

Examinons maintenant un exemple à trois et quatre serveurs.

Dix clients et trois ou quatre serveurs ($C_{10}S_{3/4}$) sur MVB

Dans cet exemple, voici les positions initiales pour les trois serveurs, et celles fixes pour les dix clients:

```
C1:(4.21568933698,13.0906866477),    // client
C2:(2.39831963666,2.55813329507),    // client
C3:(11.4988982107,5.98222726769),    // client
C4:(1.01509004785,8.61843429069),    // client
C5:(2.25268835679,4.93321261785),    // client
C6:(8.59620834077,10.4735832543),    // client
C7:(3.66716750044,6.08417985778),    // client
C8:(6.28859284254,6.3799046019),     // client
C9:(7.01790955244,0.00584793743018), // client
C10:(9.58172773178,12.0979879955),   // client
S1:(7.43595511533,12.0976233734),    // serveur
S2:(2.75603719184,8.7170832533),     // serveur
S3:(12.0182382185,12.5297384926).    // serveur
```

MVB est utilisé ici sur une grille de deux-cent-vingt-cinq (15x15) points (D_GRILLE=15) avec une maille (MAILLE) de ½. Nous avons calculé (avec les outils en 'C' vus au chapitre III) la solution globale optimale. Voici les valeurs obtenues avec trois et quatre serveurs:

- $CG^*_{10,3}$=24.34403745… (=0.0188…+11.8665…+12.4586…);
- $MM_{3\text{-}10}{}^*$={P_1,P_2,P_3}={(7,0),(3.5,6),(8.5,10.5)};
- $CG^*_{10,4}$=18.740182720461815… (=11.86…+0.0188…+6.837…+0.0178…);
- $MM_{4\text{-}10}{}^*$={P_1,P_2,P_3,P_4}={(7,0),(3.5,6),(8.5,11.5),(11.5,6)}.

La constante INV_PAS_DEPL_SRV1, qui détermine la vitesse des serveurs, vaut toujours 40 (vitesse relativement rapide), comme dans l'exemple précédent.

Avant de voir comment se comporte le système exploratoire, sur cet exemple, rappelons quelques éléments théoriques obtenus dans la partie III-A du chapitre III, et appliqués à ce dernier exemple.

La grille possède deux-cent-vingt-cinq (15x15) points. La formule *(7)* donne le nombre de positions possibles pour 'k' points (serveurs) sur la grille:

$$Card(POS_k)=D^2!/(k!(D^2-k)!); \quad (7)$$

ici, $Card(POS_k)=225!/(k!(225-k)!)$, avec $0<k\leq225$.

Si k=225, cela revient à avoir un serveur sur chaque point de la grille. Dans notre problème (C_iS_j) cela n'a de sens que si l'on a 'k' clients.
En principe le nombre de serveurs est égal au plus au nombre de clients.

Dans notre exemple, D^2 est impair (225) donc nous avons deux points d'inflexion k_1 et k_2, avec:

$$k_1=(D^2-1)/2 \text{ et } k_2=(D^2+1)/2.$$

Cela donne les valeurs suivantes: k_1=112 et k_2=113.

Si nous travaillons, comme c'est le cas présenté dans les résultats, sur une maille ½, le nombre de points de la grille est D_p, avec $D_p^2=(2D-1)^2=29^2=841$. Nous avons encore une valeur impaire pour D_p^2, ce qui nous donne deux points d'inflexion :

$k'_1=(D_p^2-1)/2=420$ et $k'_2=(D_p^2+1)/2=421$.

Dans notre exemple avec dix clients, nous n'aurons jamais plus de dix serveurs ($k \leq 10$). Ceci correspond à la phase de forte croissance de la courbe ($Card(POS_k)$), très loin du point d'inflexion (k=420).
Pour calculer les positions optimales exactes des 'k' serveurs, en théorie, plus la solution est évidente ($CG^* \rightarrow 0$ pour $k \rightarrow 10$) et plus les calculs restent coûteux. Paradoxalement la dernière position optimale (k=10) des serveurs est triviale, aucun calcul n'est nécessaire ($CG^*=0$), alors que plus on s'en approche et plus les calculs sont inaccessibles. Nous rencontrons une sorte de point de bifurcation/singularité mathématique dans la complexité des calculs.

On peut ajouter qu'une grille de deux-cent-vingt-cinq (15x15) points, avec une maille de ½, est capable de supporter (potentialité) jusqu'à huit-cent-quarante-et-un clients.

La v6 (MVB) permet de paramétrer le nombre de cycles que l'on souhaite faire faire à l'exploration. Il suffit de positionner la constante globale NB_EXPLORATIONS à la valeur choisie. Une piste d'amélioration consisterait à identifier des critères qui permettraient aux acteurs (serveurs de la nuée) de décider eux-même, sur cette base, quand il faudrait arrêter les cycles d'exploration. Cela peut être mis en place très facilement, comme cela a été déjà fait pour sortir des modes exploratoires RESTRUCT et ALEA.
Il faut jouer sur le fait que la meilleure solution traversée ne s'améliore pas au bout d'un certain nombre de cycles exploratoires effectués.
Pour préciser les choses, rappelons qu'un cycle exploratoire correspond à un retour des serveurs du mode -4 au mode RESTRUCT via la transition N27 du graphe des états internes des serveurs (cf. la figure *Fig-IV-E2-118*). C'est une étude ouverte et très intéressante à faire…

Un autre point marquant de la v6 (MVB) est le recalage du système sur la meilleure solution rencontrée pendant un cycle d'explorations. En effet les expérimentations ont bien mis en évidence un besoin de mémorisation de cette meilleure solution qui peut être traversée plusieurs fois, ou jamais. Il est impossible, pour les serveurs, de savoir en temps réel qu'ils traversent cette meilleure solution. C'est la raison du recalage en fin d'exploration (mode -2, sur la figure *Fig-IV-E2-118*), lorsque plus aucun nouvel état n'est rencontré.
On voit, sur la figure *Fig-IV-E2-121*, les graphes de plusieurs explorations avec dans chaque nœud la valeur du CG de l'état traversé. Ce sont des explorations sur notre exemple avec trois serveurs ($C_{10}S_3$). En fonction de la direction choisie par chaque serveur, lors des

Fig-IV-E2-121

déplacements, le système va dans tel ou tel état.
En rose, sur la figue *Fig-IV-E2-121*, nous avons les états optimaux traversés et en jaune les états sous-optimaux (minima locaux) traversés. En vert (29.5,25.963, 26.173…),

nous avons des états qui sont traversés plusieurs fois lors d'une même exploration.

Les points bleus, désignent des états «utiles», c.-à-d. ceux dont le CG est le meilleur jamais rencontré au cours de l'exploration courante.

Certains graphes, comme le graphe-14, ne rencontreront jamais l'état optimal (CG=24.344…), mais seulement un état sous-optimal (par exemple l'état avec CG=25.409…).
Le graphe-13 (cf. la figure *Fig-IV-E2-121*) est passé par le sous-optimal (jaune) avant de traverser l'optimal (rose). Dans ce cas le système s'est stabilisé dans un état très mauvais (CG=32.979…).

Dans les graphes graphe-12 et graphe-13 le système emprunte la même séquence d'états (27.001…/25.409…/26.173…). En sortie de cette séquence le premier traverse l'état optimal (CG=24.344…), sans s'y arrêter. Il se stabilisera beaucoup plus tard avec CG=26.983…, alors que le second graphe traverse quatre états, dont certains déjà rencontrés, avant de traverser l'état optimal et finir beaucoup plus tard dans un état très mauvais (CG=32.979…).

Ces graphes d'états traversés, montrent la dynamique d'exploration en donnant une idée des reliefs de l'espace des états.
En effet le système peut quitter un état de plusieurs façons. L'état CG=25.409… peut conduire à l'état optimal CG=24.344…, mais aussi au CG=26.173…, qui peut aussi aller vers le CG=24.344… ou vers le CG=25.963…

On voit également comment le système peut repasser par des états «connus» (CG=25.409…, CG=26.056…, CG=24.344…) et que certains états sont communs à des graphes distincts.

Toutes ces observations, sur la dynamique du système d'exploration, posent la question de l'existence d'un méta-gradient, c.-à-d. un gradient au niveau des multi-points ou de leur équivalent en coût, à travers la grandeur CG. Pour définir ce méta-gradient il faudrait caractériser la topologie de l'espace des états traversés, en dessiner les contours, les reliefs et les obstacles.

La technique naturelle du ruissellement ou de l'érosion, par opposition à artéfactuelle ou calculatoire, devrait lever le voile sur cet espace sous-jacent.

Une réponse à cette question, de l'existence d'un méta-gradient, peut trouver des pistes grâce à la «*dérive des connaissances*» qui sera abordée au chapitre V et qui a fait l'objet de ma thèse, soutenue le 15 juillet 1992 au Mans. Basée sur la gravitation informationnelle ou relationnelle, cette approche permet de renforcer les liens entre des entités en interaction.

Appliquée au méta-gradient les entités sont les états, et les liens décrivent le passage d'une entité à une autre.

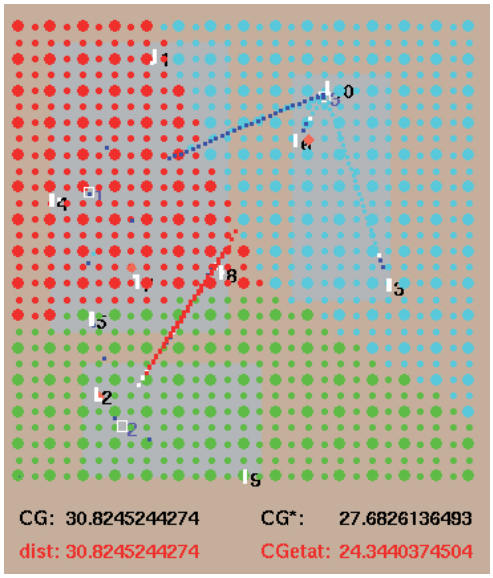


Fig-IV-E2-122 - top 2055

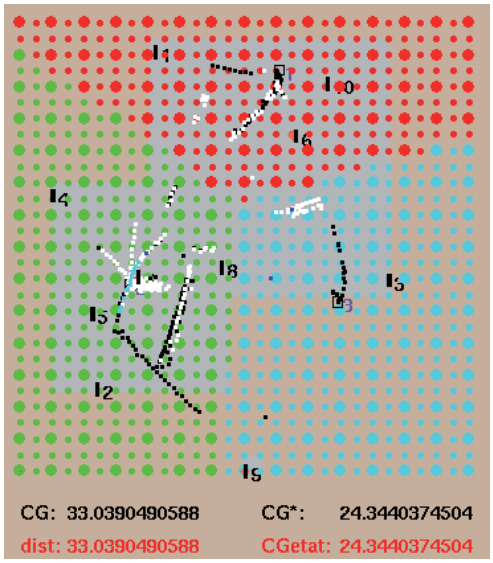


Fig-IV-E2-123 - top 2111

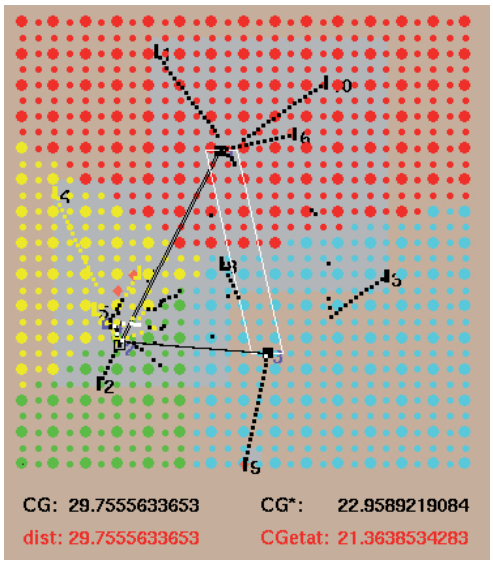


Fig-IV-E2-124 - top 2207

Première étude

Cette étude est intéressante puisqu'elle a été faite avant l'introduction des seuils EXPLO_CL_ETATS3 et EXPLO_CL_ETATS4, respectivement en charge de sortir les serveurs d'une situation (modes RESTRUCT et ALEA) où ils ne se déplacent plus et où il n'y a plus de nouvel état traversé (toujours pendant les modes RESTRUCT et ALEA).

On voit sur les graphiques des figures *Fig-IV-E2-122/127*, que les serveurs buttent sur ces deux problèmes :

- au top 2055, sur la figure *Fig-IV-E2-122*, nous sommes juste avant que les trois serveurs sortent du mode RESTRUCT, grâce au seuil EXPLO_CL_ETATS1 (requêtes résiduelles consécutives sans changement d'état). Le système a traversé trente-quatre états dont l'état optimal (CGetat=24.344…) sans s'y être arrêté. En effet la valeur CG de l'état courant est de 30.824… et la valeur optimale (CG^*) de cet état vaut 27.682… ;
- au top 2111, sur la figure *Fig-IV-E2-123*, le système possède encore trois serveurs et donc trois régions (couleurs rouge, vert et bleu sur le diagramme de Voronoï). Alors que CG=33.039… le système est arrivé en fin de son cycle d'exploration et se cale sur le meilleur état traversé (CG^*=CGetat=24.344…) ;
- au top 2190, c'est la fin du cycle d'exploration avec trois serveurs. Le meilleur état traversé de tous les cycles (CGetatK) est CG^*_3=24.344…, et le meilleur état traversé (CGetoileK), toujours de tous les cycles, vaut 24.335…. CGetoileK est meilleur que CGetatK, car le second est calculé sur la grille, alors que le premier l'est dans l'espace euclidien réel en deux dimensions. Comme nous l'avons signalé, le dernier état traversé est perdu, au sens prise en compte dans les valeurs optimales. Il s'agit ici d'un état qui vaut 34.6915…, très loin de l'optimal. Le système va donc cloner un serveur (de couleur jaune sur les graphes) ;
- au top 2207, sur la figure *Fig-IV-E2-124*, le clonage du quatrième serveur vient d'avoir lieu. Il a été créé à côté du serveur vert qui possédait

le plus de clients. Le début d'un nouveau cycle d'exploration démarre ;

- au top 9317, sur la figure *Fig-IV-E2-125*, le système, après avoir traversé seize états, s'est stabilisé dans l'état CG=20.385…, après avoir traversé CG^*_4=18.740…, le meilleur état possible (calculé) avec quatre serveurs. Le nombre de requêtes résiduelles est très faible, rendant la pente du compteur associé, très lente, et retardant d'autant, et pour rien, le dépassement du seuil EXPLO_CL_ETATS1 qui fera sortir les serveurs de l'exploration (mode RESTRUCT). La création, ultérieure à cette expérimentation, du seuil EXPLO_CL_ETATS3 accélérera cette résolution ;
- au top 17646, sur la figure *Fig-IV-E2-126*, le calage s'opère sur le meilleur état traversé qui est l'optimum CG^*_4=18.740…. Nous sommes entre la fin de l'exploration et le prochain clonage ;
- au top 18263, sur la figure *Fig-IV-E2-127*, le clonage du cinquième serveur, en noir, vient d'être réalisé (top 18241). L'opération s'est faite à proximité du serveur jaune.

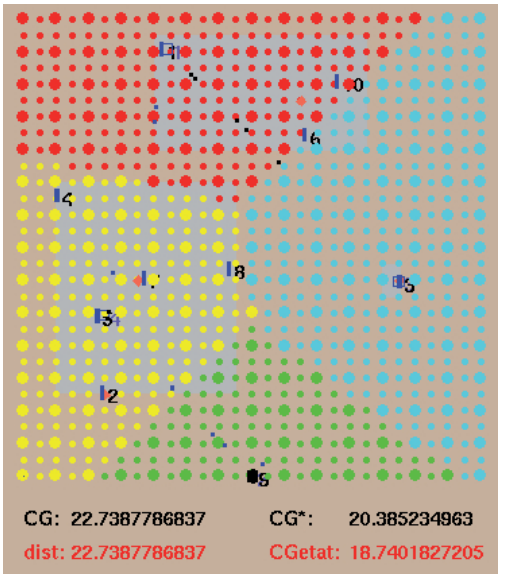


Fig-IV-E2-125 - top 9317

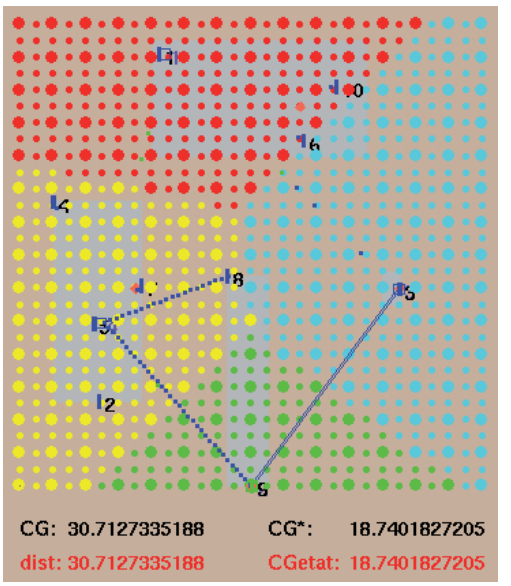


Fig-IV-E2-126 - top 17646

Deuxième étude

Les quatre seuils (EXPLO_CL_ETATS1, EXPLO_CL_ETATS1_1, EXPLO_CL_ETATS3 et EXPLO_CL_ETATS4) ont été implantés. L'expérimentation est configurée pour faire trois explorations à chaque niveau ('k', étant le nombre de serveurs). Elle montre le bon fonctionnement de ces quatres seuils :

- au top 2905, les trois serveurs viennent de quitter le mode RESTRUCT. Le système a traversé trente-quatre états. Les compteurs liés aux seuils EXPLO_CL_ETATS1 et EXPLO_CL_ETATS3 ont servi pour sortir de ce mode. Dès la première exploration le système a traversé CG^*_3 ;
- au top 3154, les trois serveurs quittent le cycle d'exploration pour se caler sur CGetat (=CG^*_3). Ils font chacun leur descente de gradient dans leur région respective pour obtenir un CG=24.309… inférieur à

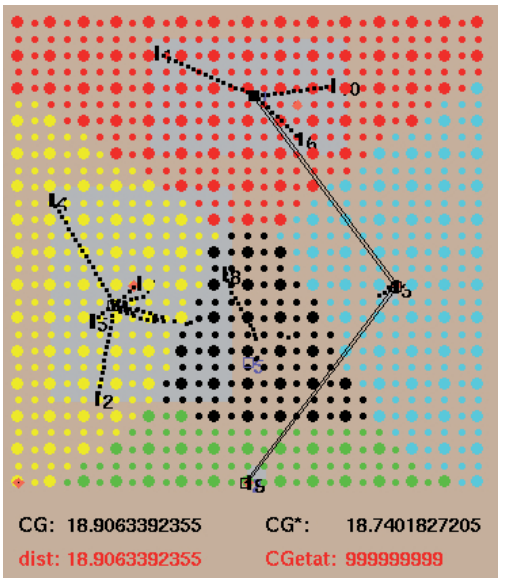


Fig-IV-E2-127 - top 18263

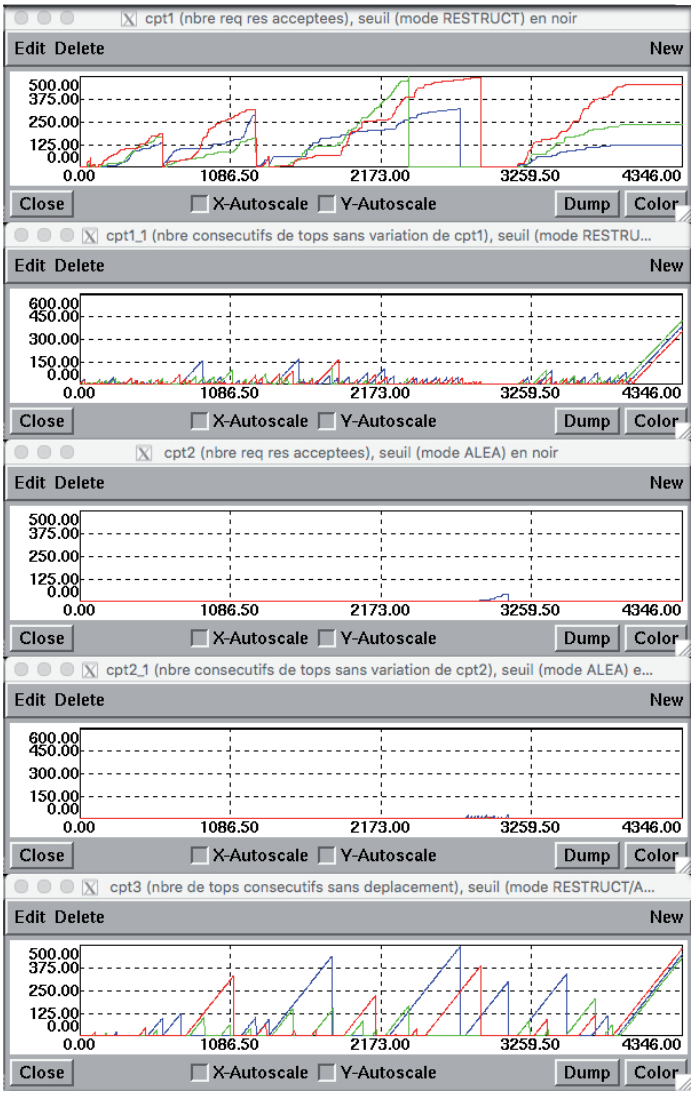


Fig-IV-E2-128 – top 4347

l'optimal théorique (CG^*_3). Le système commence une deuxième exploration avec ses trois serveurs, qui ne donnera rien de mieux, puisque l'optimum a déjà été atteint ;

- au top 3180, c'est le début de la deuxième exploration ;
- au top 4347 (cf. la figure *Fig-IV-E2-128*), le système est figé dans un attracteur. Le compteur lié au seuil EXPLO_CL_ETATS1 ne joue pas son rôle, puisque sa valeur est constante, consécutivement à l'arrêt d'arrivée de requêtes résiduelles. Cette situation est partagée par les trois serveurs. Comme on peut le voir sur la figure *Fig-IV-E2-128*, ce sont les compteurs (cpt1_1 et cpt3), liés à deux autres seuils, qui prennent le relais pour sortir les serveurs du mode RESTRUCT, finalement sur le manque de mobilité (cpt3). Le système a exploré trente-sept états depuis le départ. CGetat est toujours à l'optimal (CG^*_3), alors que le système est dans un état non-optimal ;
- au top 4494, c'est la fin de la deuxième exploration avec trois serveurs ;
- au top 5761, c'est la fin de la troisième exploration et le clonage d'un quatrième serveur. Le système a exploré trente-neuf états, dont deux nouveaux (28.294… et 26.721…) ;
- au top 5783, c'est le début de la première exploration avec quatre serveurs ;
- au top 8859, c'est la fin de la première exploration avec quatre serveurs. Le système a déjà atteint l'optimal (CG^*_4=18.740…) ;
- au top 9033, le système entame la deuxième exploration ;
- au top 11648, c'est la fin de la troisième exploration, après vingt-sept états traversés. Le système va cloner un cinquième serveur ;
- au top 12216, nous sommes dans la première exploration avec cinq serveurs. Le premier compteur (cpt1) est constant depuis un certain temps pour les cinq serveurs. C'est le troisième compteur (cpt3) qui va débloquer tout le monde, par manque de mobilité des serveurs qui sont en phase d'exploration ;
- au top 16003, nous sommes dans la deuxième exploration juste avant le calage. CGetat et CG^* valent 14.038… ;
- au top 16578, le serveur rouge est revenu seul dans le mode RESTRUCT. Il bloque les autres pour sortir du cycle d'exploration. Ne bougeant plus, le compteur cpt3 va l'en sortir ;
- au top 18202, c'est la fin de la troisième exploration et un recalage sur CGetat=14.038…, avec, après une dernière descente de gradient, un CG=13.823… ;
- au top 18205, le système a parcouru vingt-deux états en

tout. Il clone un sixième serveur;
- au top 19469, c'est la fin de la première exploration. Le système a traversé six états pour atteindre CGetatK=10.334… et CGetoileK=10.119…;
- au top 21842, c'est la fin de la deuxième exploration. Le système a traversé deux nouveaux états sans changer les valeurs (CGetatK…).

On peut déjà remarquer, sur cet exemple, que plus le nombre 'k' de serveurs augmente et moins le système rencontre de nouveaux états.

Troisième étude complète

Dans cette troisième étude nous allons détailler les principaux mécanismes en œuvre dans une exploration complète, c.-à-d. pour 'k' allant de 1 à 10 (le nombre de clients), mais avec une seule exploration (NB_EXPLORATIONS=1) par profondeur 'k'. Notre exemple, bien que très long à calculer de façon exacte, n'en n'est pas moins relativement simple. En effet nous observons que, la plupart du temps, les optima sont trouvés dès la première exploration, rendant ainsi inutiles les suivantes. Il faut prendre cette remarque avec précaution, car une exploration n'est jamais inutile. Comme nous l'avons montré sur un exemple (cf. la figure *Fig-IV-E2-121*), le système apprend des choses sur l'espace des états en faisant des explorations successives. Trouver l'optimum n'est pas le seul objectif des explorations. Nous aurions dû revenir en détail sur cette notion dans les parties suivantes, après avoir introduit «*La dérive des connaissances*», au chapitre V.

Nous présentons les résultats sous forme synthétique (cf. le tableau ci-dessous), avant de regarder en détails comment les choses se sont faites:

k	3	4	5	6	7	8	9	10
top fin explo (mode -4)	1467	3401	5173	6957	8308	10403	11669	12945
CG^* (calculé en 'C')	24.344…	18.740…	-	-	-	-	-	0
CGetatK	24.344…	18.740…	14.038…	10.334…	6.827…	4.476…	2.802…	1.420…
CGetoileK	24.304…	18.750…	13.832…	10.053…	6.324…	4.056…	2.102…	0.446…
nb d'états rencontrés	22	26	22	15	2	1	1	1

Ces résultats confirment que plus le système explore en profondeur et moins il parcourt d'états. C'est assez intuitif car la solution limite, et non plausible, avec autant de serveurs que de clients (ici dix), consiste à mettre un serveur le plus près possible de chaque client. Il en découle un CG (coût) très faible. Quand on approche cette solution limite, dès k=8, il suffit de rajouter un serveur à proximité du serveur qui a plusieurs clients. C'est d'ailleurs la stratégie qui a été choisie dans MVB. Se faisant un seul état est nécessaire pour tomber dans l'optimum suivant (k+1). On pourrait penser, néanmoins, que cela n'empêcherait pas le système d'explorer l'espace des états.

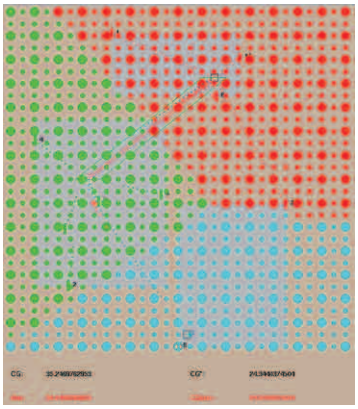


Fig-IV-E2-129 - top 1343

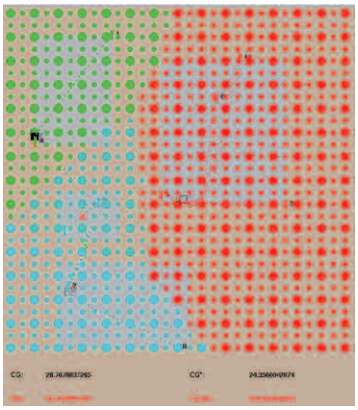


Fig-IV-E2-130 - top 1500

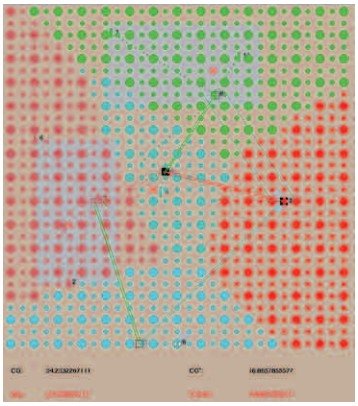


Fig-IV-E2-131 - top 3438

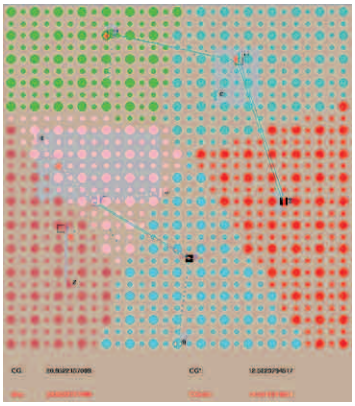


Fig-IV-E2-132 - top 5198

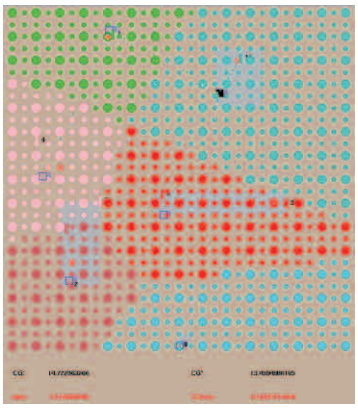


Fig-IV-E2-133 - top 6705

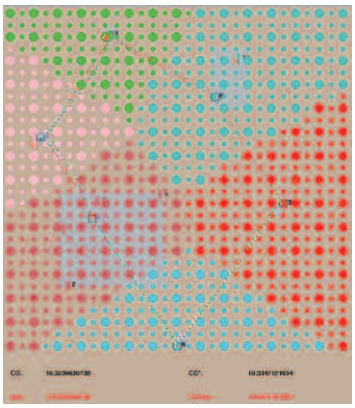


Fig-IV-E2-134 - top 6952

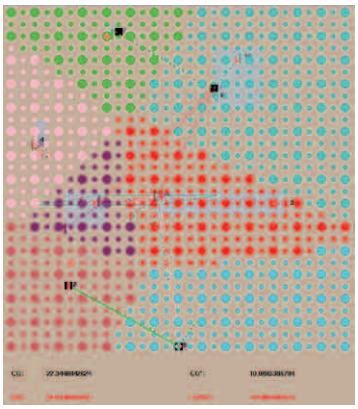


Fig-IV-E2-135 - top 6987

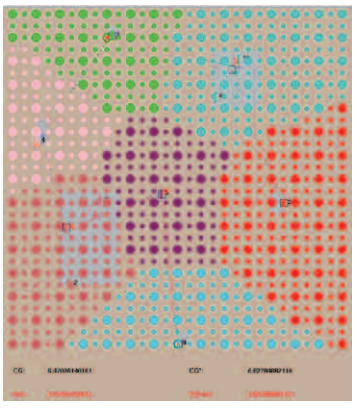


Fig-IV-E2-136 - top 8163

En pratique, dans les phases d'exploration (RESTRUCT et ALEA), les serveurs se déplacent uniquement à l'intérieur de leur propre rectangle de recouvrement, défini par les positions de leurs clients (*cls_dom*). Dans ces conditions, plus le nombre de clients d'un serveur se réduit et plus ce dernier se rapproche du ou des clients de sa région ; ce faisant, plus le rectangle de recouvrement diminue en surface autour de lui.

Cela est particulièrement significatif quand 'k' augmente. En effet si un serveur ne possède plus qu'un client, son rectangle de recouvrement, et donc d'exploration, se réduit considérablement pour être cantonné au strict voisinage immédiat du client en question. Plus ce rectangle est petit, en surface, et moins le serveur va explorer (bouger).

Ceci explique la baisse du nombre d'états traversés en fonction de l'augmentation de 'k' (la profondeur d'exploration). Cela participe à une forme d'auto-régulation de l'exploration.

Ne connaissant pas les valeurs théoriques (calcul en 'C') pour k>4, car trop coûteuses en calculs pour notre machine, nous ne pouvons pas aller plus loin sur la capacité de l'exploration à suivre les optima exacts.

On sait que le système a trouvé ceux pour k=3 et k=4. En s'appuyant sur la remarque précédente (réduction de la surface des rectangles de recouvrement avec la profondeur 'k' de l'exploration qui augmente) on peut penser que les résultats obtenus avec k>4 ne seront pas loin de l'optimum. L'incertitude baisse, de toute façon, avec l'augmentation de cette profondeur.

Notons que l'expérimentation a été faite avec une seule exploration par niveau 'k' et qu'elle pourrait être faite avec davantage. Reste à trouver une machine capable de faire ces calculs exacts…

Sur les figures *Fig-IV-E2-129/144*, nous voyons l'évolution de la résolution sur les huit niveaux de profondeur allant de k=3 à k=10. Les diagrammes de Voronoï affichent une couleur différente par région.

Les zones en gris clair représentent les rectangles de recouvrement des serveurs. Ce sont les fameuses zones dans lesquelles l'exploration est faite en modes RESTRUCT et ALEA. Ils recouvrent les clients de chaque serveur (région), indépendamment de la position des serveurs eux-mêmes. Ce calcul de zone est fait localement par chaque serveur aux changements d'état.

Commentons rapidement ces différents diagrammes :

- au top 1343, sur la figure *Fig-IV-E2-129*, le système est en fin d'exploration avec trois serveurs. Il a traversé l'optimum (CG^*_3) et va cloner un quatrième serveurs à proximité du serveur vert qui possède le plus grand nombre de clients ;
- au top 1500, sur la figure *Fig-IV-E2-130*, le système vient juste de créer ce quatrième serveur. Sa région n'apparaît pas encore sur le diagramme de Voronoï. Il s'intercale entre la position du serveur vert et celle du serveur bleu. Les valeurs de CG… affichées ne sont pas encore significatives ;
- au top 3438, sur la figure *Fig-IV-E2-131*, la zone en violet clair, pour ce quatrième serveur, est apparue. Nous sommes dans la même situation qu'au top 1500, car un cinquième serveur vient d'être créé, mais sa région n'apparaît pas encore (elle n'a pas encore été calculée) dans le diagramme de Voronoï, bien qu'on puisse assez facilement la deviner (la zone bleu s'étirant en deux zones vers le haut de la grille). On voit déjà la surface totale des rectangles de recouvrement (en gris) se réduire. Les valeurs CG… ne sont pas encore bonnes, l'exploration commence ;
- au top 5198, sur la figure *Fig-IV-E2-132*, le système a cloné un sixième serveur, que l'on voit très bien sur le diagramme de Voronoï. L'exploration n'est pas terminée car CGetat n'est pas à sa meilleure valeur. Trois serveurs (rouge, bleu du bas de la grille et vert) n'ont qu'un client chacun. Les serveurs bleu (haut de grille) et framboise ont

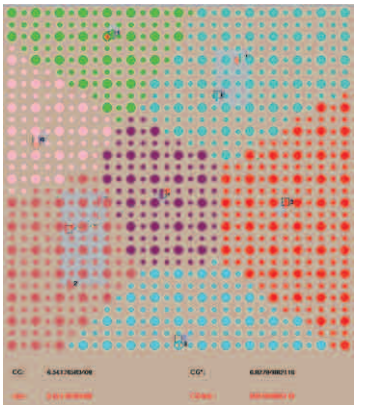


Fig-IV-E2-137 - top 8303

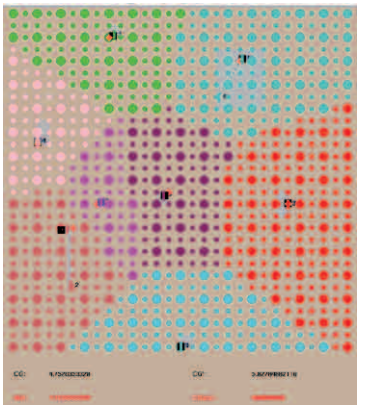


Fig-IV-E2-138 - top 8352

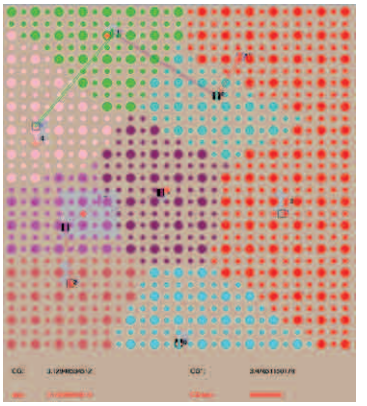


Fig-IV-E2-139 - top 10440

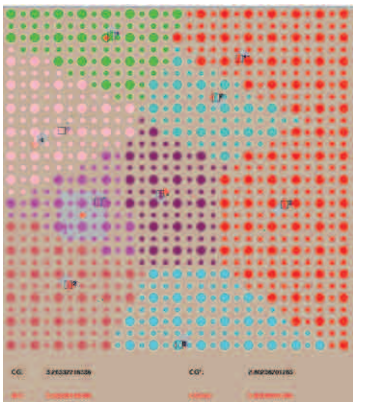


Fig-IV-E2-140 - top 11647

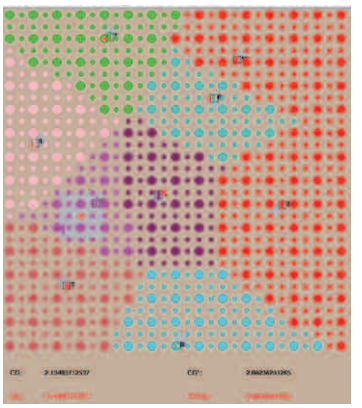


Fig-IV-E2-141 - top 11666

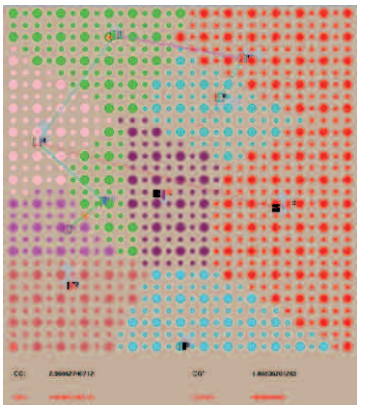


Fig-IV-E2-142 - top 11707

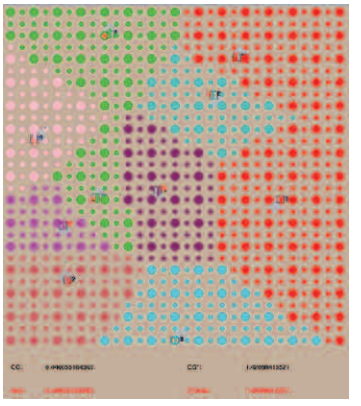


Fig-IV-E2-143 - top 12708

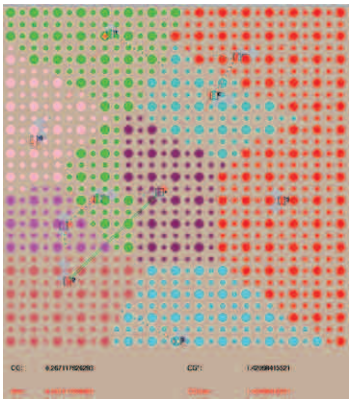


Fig-IV-E2-144 - top 12955

deux clients chacun. Seul le serveur rose possède trois clients;

- au top 6705, sur la figure *Fig-IV-E2-133*, le système a traversé son meilleur état (CGetat=10.334…) avec ses six serveurs, sans s'y arrêter. Quatre serveurs ont deux clients chacun;
- au top 6952, sur la figure *Fig-IV-E2-134*, le système vient de terminer son exploration à six serveurs, et se cale sur la meilleure solution rencontrée (CGetat=CG^*=10.334…). Cette solution comporte la région framboise avec quatre clients, la bleue (du haut de la grille) avec deux clients et les autres avec un seul client chacune. On voit se dessiner la suite de l'exploration qui va séparer progressivement cette région framboise en clonant les serveurs;
- au top 6987, sur la figure *Fig-IV-E2-135*, effectivement le septième serveur violet arrive sur la région framboise. Seules deux régions sont impactées par l'arrivée de la septième région;
- au top 8163, sur la figure *Fig-IV-E2-136*, le système se cale sur la meilleure solution (CGetat=CG^*=6.827…) traversée avec sept serveurs. Il réalise une ultime descente de gradient (pas encore terminée) qui le conduit à une solution meilleure (CG=6.470…). Comme prévu, le septième serveur violet a pris un client au serveur framboise, qui n'en n'a plus que trois;
- au top 8303, sur la figure *Fig-IV-E2-137*, c'est la fin de l'exploration sans changement notable;
- au top 8352, sur la figure *Fig-IV-E2-138*, le système vient de cloner un huitième serveur violet clair à côté du serveur framboise qui possède le plus grand nombre de clients. La surface totale des rectangles de recouvrement diminue toujours au rythme des nouveaux serveurs arrivants. La topologie générale du diagramme de Voronoï bouge très peu;
- au top 10440, sur la figure *Fig-IV-E2-139*, un neuvième serveur vient d'être créé en rouge à côté du serveur bleu (haut de la grille). En effet ce serveur bleu avait deux clients comme le serveur framboise. C'est lui qui a été «choisi»;
- au top 11647, sur la figure *Fig-IV-E2-140*, le système se cale sur sa meilleure solution (CG^*=CGetat=2.802…) avec ses neuf serveurs. Il ne reste plus qu'un serveur (violet clair) qui possède deux clients;
- au top 11666, sur la figure *Fig-IV-E2-141*, le système continue sa descente de gradient qui le conduit vers une meilleure solution (CG=2.134…) provisoire. Le diagramme de Voronoï n'évolue plus beaucoup;
- au top 11707, sur la figure *Fig-IV-E2-142*, le dixième et dernier serveur (vert - partie centrale de la grille) vient d'être créé à proximité du serveur violet clair. Nous sommes très proche de la version finale. Chaque serveur possède maintenant un seul client;
- au top 12708, sur la figure *Fig-IV-E2-143*, l'exploration se cale sur la meilleure solution trouvée (une seule) avec CG^*=CGetat=1.420… et effectue une descente de gradient qui l'emmène à CG=0.267… L'écart important s'explique ici par la précision des calculs exacts sur la grille (maille), en comparaison avec une descente de gradient, sur un espace euclidien réel en deux

dimensions, avec un cœfficient utilisé pour cette descente valant 0.1*cosα;
- au top 12955, sur la figure *Fig-IV-E2-144*, il n'y a pas de changement notoire.

Pour conclure sur cette troisième étude, on peut commenter les quelques graphiques obtenus au top 12986, c.-à-d. à la fin de toutes les explorations.
Le premier graphique trace l'évolution des variables «partagées» CGetoileK (courbe verte sur la figure *Fig-IV-E2-145*) et CGetatK (courbe bleue sur la figure *Fig-IV-E2-145*) qui représentent respectivement la meilleure solution effective traversée pendant les explorations, et la meilleure solution calculée à chaque niveau de profondeur 'k'.
Globalement l'écart entre la valeur calculée (toujours supérieure) et la valeur effective croît avec la profondeur 'k'. Ceci est principalement dû au mode de calcul du déplacement dans l'espace euclidien et à la maille choisie pour la grille. Plus cette maille est fine (faible valeur) et moins l'écart est perceptible.

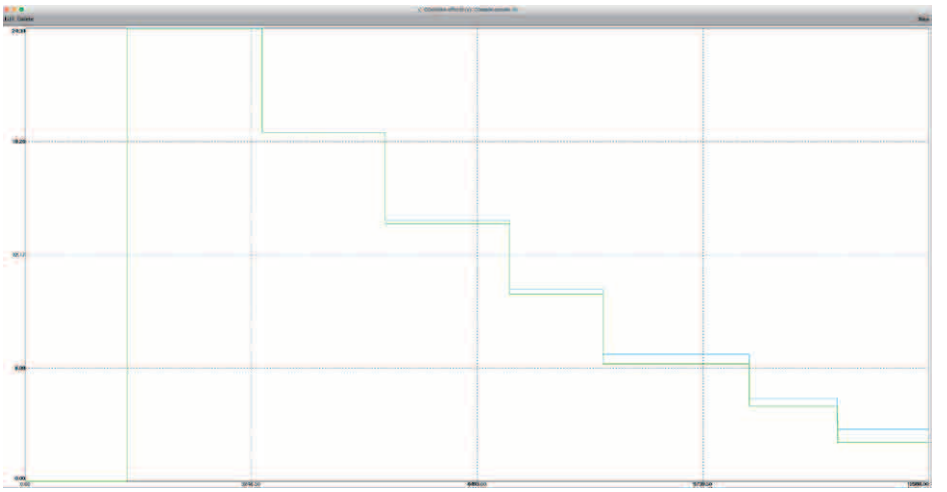


Fig-IV-E2-145

Le deuxième graphique (cf. la figure *Fig-IV-E2-146*) montre l'évolution à chaque top des variables «partagées» CG (en rouge), CG* (en vert) et CGetat (en bleu).
La courbe rouge affiche les valeurs traversées lors des explorations. Les nombreux pics verticaux représentent des explorations dans des zones des états éloignées de leurs puits (optima locaux), soit dans le cadre de l'algorithme de contraction/expansion (mode RESTRUCT), soit en mode ALEA ou encore RAZPOS. Les descentes «progressives», de la courbe rouge, montrent les descentes de gradient opérées par les serveurs.
La courbe verte fonctionne en paliers qui correspondent à chaque état traversé et son optimum calculé sur la grille par les serveurs. En principe cette courbe est en-dessous de la rouge sauf lorsque les serveurs font des descentes de gradient (mode GRADIENT2) dans l'état courant.
La partie basse de la courbe rouge est corrélée avec la courbe verte.

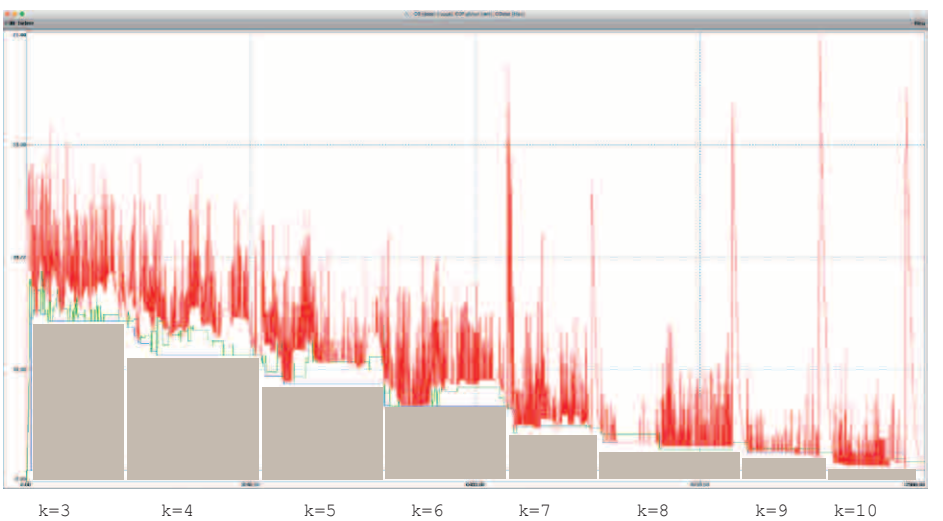


Fig-IV-E2-146

La courbe bleue visualise les grands paliers liés à chaque niveau ‘k’ de profondeur des explorations. Elle est généralement sous la verte, sauf lorsque le meilleur état est traversé en fin de cycle d’exploration pour un niveau ‘k’ donné. C’est par exemple le cas autour du top 9739 sur la figure *Fig-IV-E2-146*.

Le diagramme, sur la figure *Fig-IV-E2-147*, montre l’évolution des périodes d’observation (PO) de chaque serveur pendant les cycles exploratoires. Les courbes, une par serveur, donnent une idée du niveau de coopération entre certains serveurs au sein des explorations. Lorsque des serveurs interagissent via des requêtes résiduelles et qu’ils atteignent une certaine stabilité, ils échangent entre eux leur propre période d’observation afin de synchroniser leur traitement des échecs via ces requêtes résiduelles. Nous avons traité ce sujet dans les parties IV-B4 et IV-B5. Ces courbes nous montrent les serveurs qui ont coopéré pendant les explorations. Des groupes sont bien dessinés dans les premiers niveaux d’exploration. En fin d’exploration, dans les niveaux à partir de k=8, chaque serveur devient plus autonome et ne partage plus sa PO. Elles deviennent toutes non-correlées.

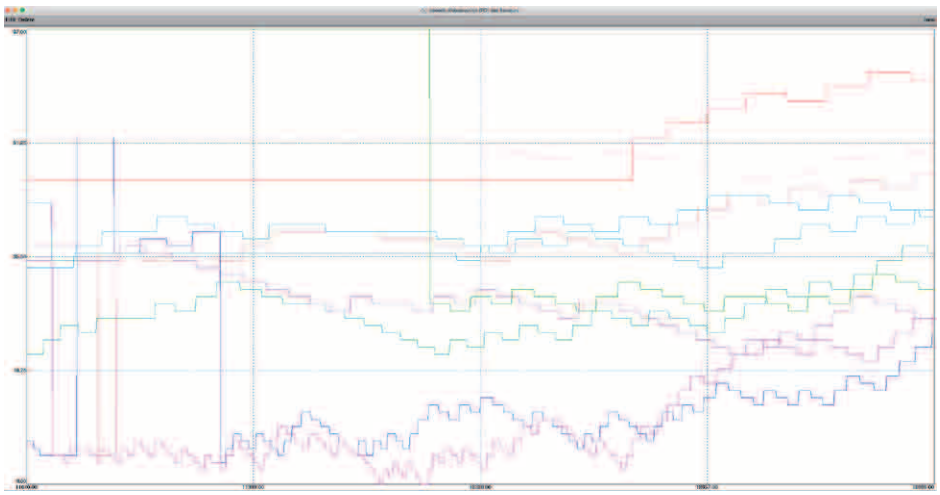


Fig-IV-E2-147

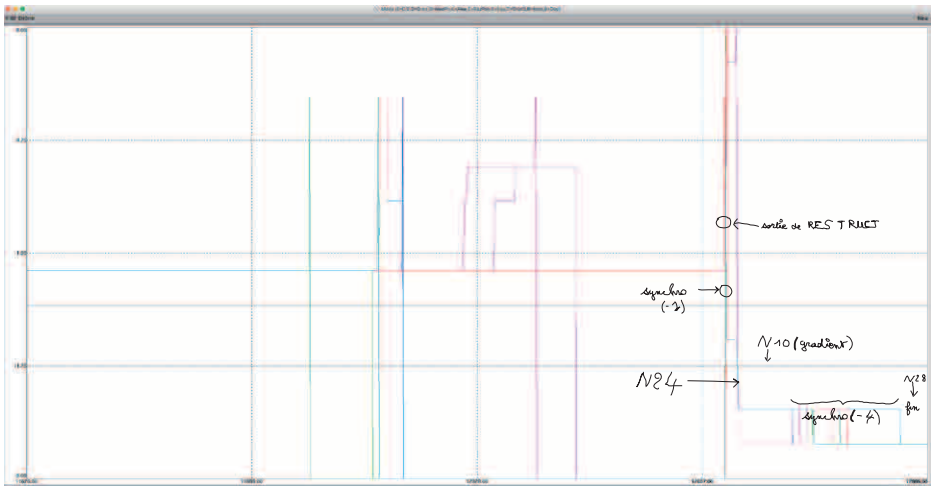


Fig-IV-E2-148

Le dernier diagramme, sur la figure *Fig-IV-E2-148*, montre les changements de mode au sein de chaque serveur à la fin des explorations. Tous les serveurs sont autonomes et disposent du même moteur (graphe d’états internes), qu’ils utilisent de manière asynchrone entre eux. Néanmoins, grâce à la notion de données «partagées», ils peuvent créer des points de rendez-vous (à l’image de l’opération ‘Z’ dans les sémaphores sur *Linux*). Cela leur permet de se synchroniser quand cela est nécessaire. On voit un premier point de synchronisation, pointé sur la figure *Fig-IV-E2-148*, avec le mode -1. Nous sommes en fin de dernière exploration à la profondeur maximale (k=10). Tous les serveurs ont trouvé l’équilibre et ont réalisé une descente de gradient dans la région de l’état courant, où ils se trouvent. Les uns après les autres ils passent dans le mode -1. Dès que l’un d’entre eux y arrive, un compteur «partagé» est

incrémenté. Tant que le dernier n'y est pas encore arrivé, les autres restent dans ce mode de rendez-vous.
Dès que le dernier y arrive, le compteur, testé par tous à chaque top, libère tous les serveurs pour qu'ils passent au mode -2, et se calent sur la meilleure dernière solution trouvée, avant de passer finalement au mode -3.
Arrivés dans ce mode -3 chaque serveur fait une ultime descente de gradient (N10) pour optimiser la meilleure solution trouvée.
On voit, sur la partie en bas à droite de la figure *Fig-IV-E2-148*, comment chaque serveur passe (N24) en mode -4 dès qu'il a fini sa descente de gradient.
Ce dernier mode est aussi un point de synchronisation entre les serveurs. Il y a trois possibilités en sortie de ce rendez-vous. Soit tous les serveurs repartent (N27) dans une nouvelle exploration au même niveau ('k' constant), soit on clone un serveur avant de repartir (N26) pour une nouvelle série d'explorations ou encore, et c'est ce que l'on voit sur la figure *Fig-IV-E2-148*, c'est la fin définitive des explorations (N28).

Étude de la vitesse de déplacement sur la résolution, dans MVB

Nous avons vu l'importance du déplacement des serveurs dans le processus de résolution exploratoire. Ce processus est fondé sur l'algorithme de contraction/expansion, mais aussi sur les déplacements des serveurs vers des positions aléatoires à l'intérieur d'un cadre (les régions propres à chaque serveur).
Pour lancer une première étude de l'impact de la nature de la vitesse de déplacement des serveurs dans ce processus de résolution exploratoire, nous allons considérer trois cas emblématiques qui donnent des résultats assez voisins:

- a) *vitesse constante*: dans ce cas tous les serveurs de la simulation partagent dans le temps une même vitesse constante choisie au départ de la résolution. La valeur de cette vitesse constante, définie par la constante INV_PAS_DEPL_SRV1, doit être choisie en fonction des caractéristiques de la simulation (distances entre les serveurs et les clients, vitesse des requêtes,…). Dans les résultats présentés ci-dessous, cette constante INV_PAS_DEPL_SRV1 vaut 40. Plus cette valeur est grande et plus les serveurs se déplacent lentement vers leur cible;

k	3	4	5	6	7	8	9	10
top fin explo (mode -4)	1902	3859	5260	7331	9154	10209	11220	12538
CG* (calculé en 'C')	24.344…	18.740…	-	-	-	-	-	0
CGetatK	24.344…	18.740…	15.036…	11.529…	6.827…	4.476…	2.802…	1.420…
CGetoileK	24.270…	18.747…	14.969…	11.173…	6.276…	3.979…	2.125…	0.398…
nb d'états rencontrés	19	12	6	5	6	1	1	1

- b) *vitesse variable mais partagée*: dans ce cas la vitesse des serveurs est toujours la même pour tous les serveurs, mais elle peut changer dans le temps. Dans les résultats présentés ci-dessous, la vitesse

est indexée sur le nombre d'états (solutions) trouvés pour une profondeur 'k' donnée, pendant la résolution. Là encore plus le nombre d'états trouvés est grand et moins les serveurs se rapprocheront rapidement de leur cible. On peut remarquer que pour k=3, le nombre d'états rencontrés vaut quarante en fin de résolution. On retrouve la valeur choisie dans le cas a) pour la constante INV_PAS_DEPL_SRV1, qui fonctionne pour cet exemple. Ensuite ce nombre va progressivement décroître. Or plus cette valeur est faible et plus le serveur se déplace rapidement vers sa cible. Comme nous allons le montrer dans l'épilogue, partie IV-E3, intitulée «Vers un placement crypté de nuées de serveurs», plus les déplacements sont rapides et moins nous rencontrons de nouvelles solutions. Or nous savons également que plus la profondeur ('k') augmente et moins il y a de solutions pertinentes. Il y a donc une cohérence à accélérer la vitesse avec l'augmentation de la profondeur ('k') de l'exploration. Les résultats présentés précédemment (troisième étude complète) ont été obtenus avec la vitesse indexée sur le nombre d'états rencontrés avec un scénario similaire (22-26 états au début de l'exploration);

k	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
top fin explo (mode -4)	2400	4427	6023	7907	9226	10517	11766	13143
CG* (calculé en 'C')	24.344…	18.740…	-	-	-	-	-	0
CGetatK	24.344…	18.740…	15.036…	10.334…	6.827…	4.476…	2.802…	1.420…
CGetoileK	24.303…	18.765…	14.940…	10.054…	6.222…	3.874…	2.174…	0.422…
nb d'états rencontrés	40	31	15	15	4	3	1	1

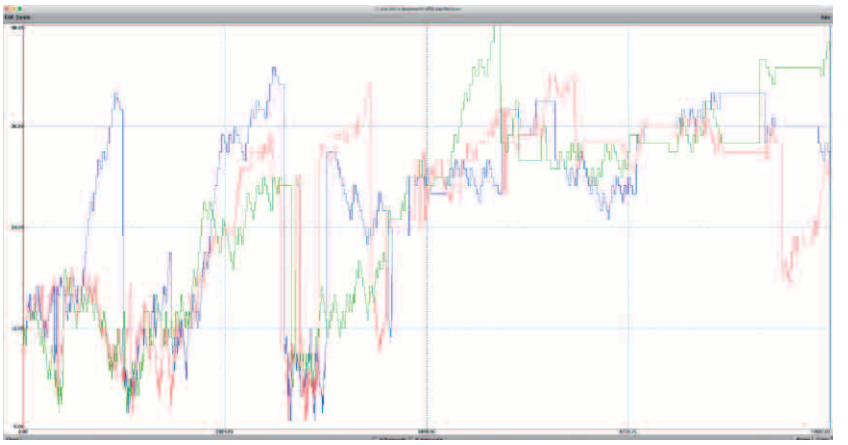
- c) *vitesse variable et propre à chaque serveur*: dans ce dernier cas chaque serveur a une vitesse variable qui lui est propre, pendant l'exploration. Pour réaliser cela nous choisissons d'indexer la vitesse de chaque serveur sur sa période d'observation (PO) qui évolue avec le temps au gré des interactions. Cela permet également de voir comment les coopérations entre serveurs, pendant lesquelles ces derniers échangent et partagent leur PO respectives, peuvent jouer dans le processus de résolution Il est à noter que la valeur des PO (calculées par les serveurs eux-mêmes en partant de 1) se retrouve, pour une majorité de serveurs, en fin d'exploration et à chaque niveau de profondeur 'k', autour de la valeur 40, celle-là même utilisée dans le cas a).

k	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
top fin explo (mode -4)	1418	3602	5127	6893	8855	10118	10920	11661
CG* (calculé en 'C')	24.344…	18.740…	-	-	-	-	-	0
CGetatK	25.409…	18.740…	15.036…	10.334…	6.827…	4.476…	2.802…	1.420…
CGetoileK	25.392…	18.766…	15.025…	10.059…	6.272…	3.969…	2.218…	0.487…
nb d'états rencontrés	30	22	16	16	4	1	1	1

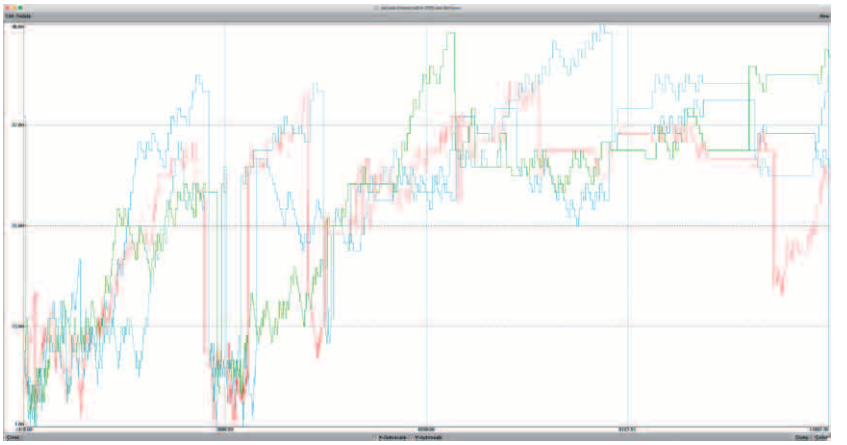
Cela permet de faire converger la simulation dans les mêmes ordres de grandeur, auto-calculés, qu'avec une vitesse constante (a) et INV_PAS_DEPL_SRV1=40 (b). La vitesse

de convergence semble plus rapide (top=11661) qu'avec une vitesse constante (top=12538) et une vitesse indexée sur le nombre d'états (top=13143), à toutes les profondeurs ('k') traversées.

Les figures *Fig-IV-E2-149/156* montrent, pour chaque niveau 'k' de profondeur, l'évolution des PO des serveurs existants. Sur la première figure on ne voit que les courbes des trois serveurs existants pour la première exploration (k=3). Ces courbes sont tracées jusqu'à la fin de la simulation. Sur la deuxième figure *Fig-IV-E2-150* démarre les courbes lors de l'exploration avec quatre serveurs, jusqu'à la fin, et ainsi de suite.

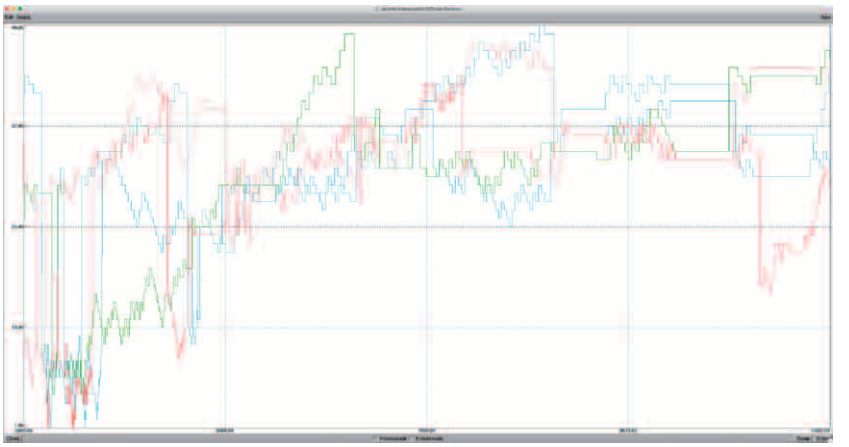


k=3 Fig-IV-E2-149



k=4 Fig-IV-E2-150

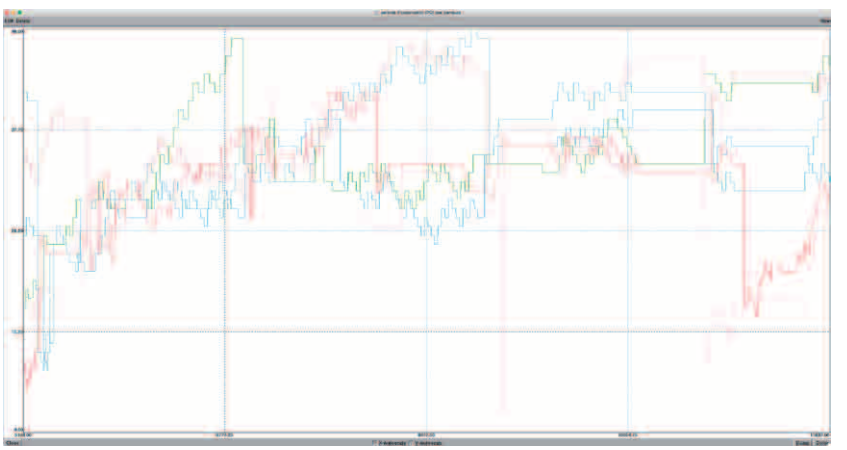
On voit nettement que la valeur moyenne des PO des serveurs, qui coopèrent, se définit autour de 40. On voit également que plus la profondeur ('k') de l'exploration augmente et moins les serveurs coopérent et donc partagent la valeur de leur PO. Il apparaît alors une plus grande dispersion des PO et donc des vitesses des serveurs entre eux. En effet la recherche de nouveaux états devenant moins prégnante avec 'k' augmentant, une forme de synchronisation des serveurs dans leur vitesse d'exploration de l'espace des solutions, n'a plus de réel intérêt.



k=5 Fig-IV-E2-151

À ce stade de l'étude, il est difficile d'en tirer des conclusions définitives.

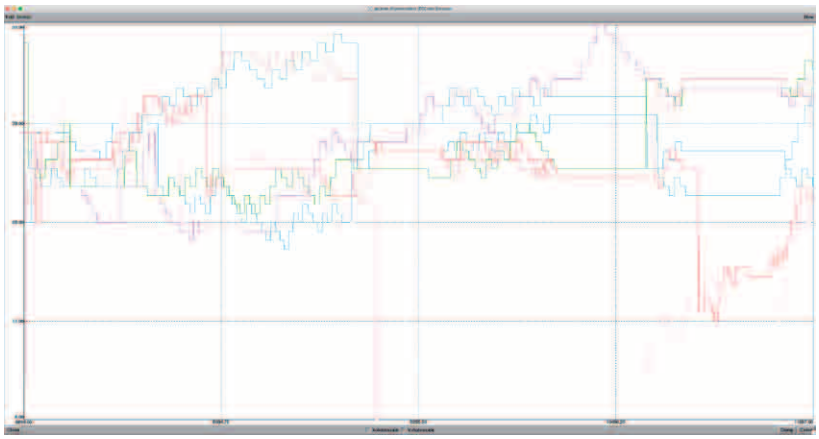
Disons quand même que la voie b), c.-à-d. avec une vitesse partagée et indexée sur le nombre d'états traversés, paraît être une bonne approche qualitative de l'exploration.



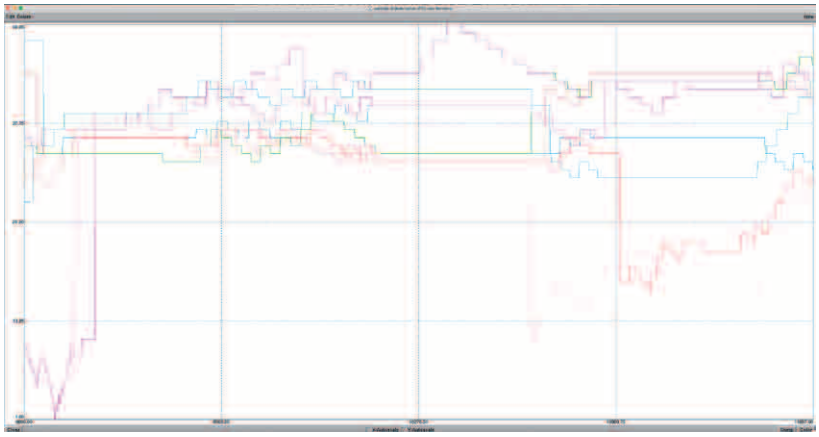
k=6 Fig-IV-E2-152

Le chiffre 40 n'apparaît plus ni par hasard, ni comme le fruit d'un tâtonnement, mais comme un ordre de grandeur qui s'impose

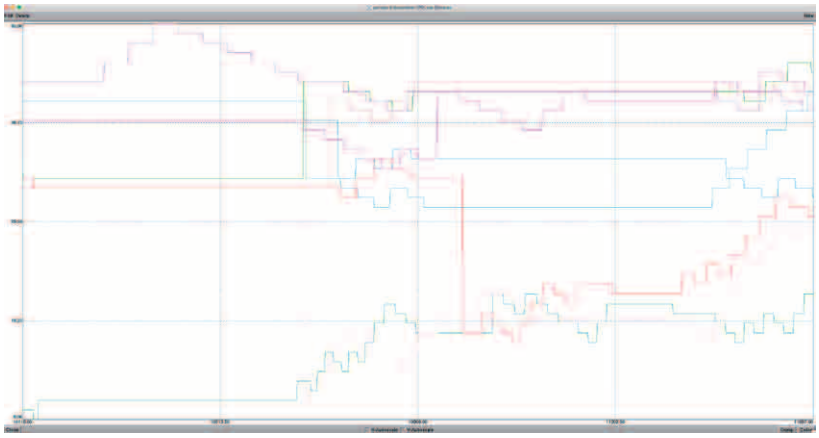
naturellement par la nature du problème à résoudre. À suivre…



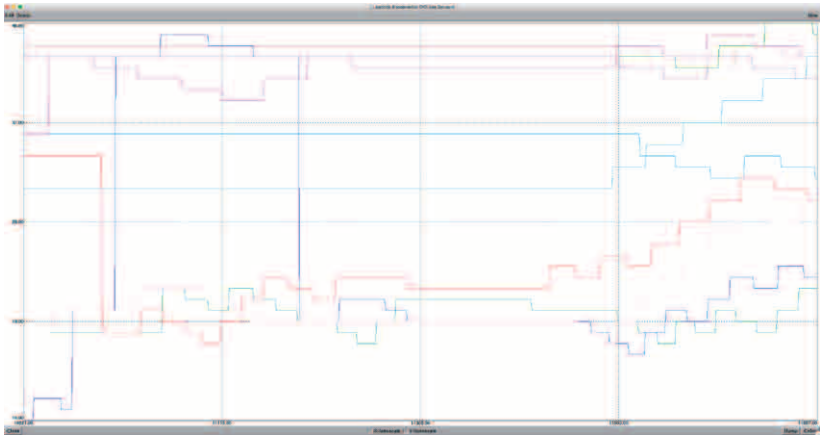
k=7 Fig-IV-E2-153



k=8 Fig-IV-E2-154



k=9 Fig-IV-E2-155



k=10 Fig-IV-E2-156

IV-E3) Épilogue

Nous pouvons résumer nos travaux en trois points :

- le premier est une méthode de résolution (MVB) du problème (C_nS_k) de l'allocation dynamique de ressources partagées. Grâce à notre approche heuristique, fondée sur une étude approfondie du problème (dynamique, gradients,…), nous avons pu obtenir (aux chapitres III et IV) des résolutions fines et non coûteuses en temps de calcul, basées sur l'exploration sans apprentissage de l'espace des solutions via un processus autopoïétique, sans avoir recours à des calculs exhaustifs ;
- le deuxième réside dans des perspectives à cette approche dans plusieurs domaines comme celui de la représentation des connaissances (en passant par l'étude des lettres de l'alphabet latin), ou encore celui du placement crypté de nuées de serveurs de données ;
- le troisième et dernier point réunit toutes ces perspectives dans un principe «philosophique» qui fait l'éloge de la lenteur[10]. Ce principe peut être considéré comme l'un des résultats le plus intéressant à nos yeux de tous ces travaux, dans la mesure où il a une portée qui les dépasse largement et qu'il n'était absolument pas recherché au départ (sérendipité). Il constitue le fil rouge de notre quête incarnée par nos derniers résultats obtenus dans la projection de MVB vers un monde distribué et ouvert non simulé (Internet), mais aussi, et de façon beaucoup plus surprenante dans notre expérience artistique avec l'encre.

Résolution autopoïétique

Notre processus de résolution, autopoïétique, et continu, est piloté par des interactions entre des entités autonomes impliquées dans une résolution collective et auto-organisée. Il produit une membrane en stabilisant le système (c.-à-d. ses acteurs dans un contexte donné et possiblement évolutif) dans un état de résolution en appliquant un principe d'optimisation de l'usage des ressources en jeu. Aucune intervention exogène n'est nécessaire pour déterminer quand la membrane est formée. Toujours à l'image des nuées d'oiseaux, seule une observation de sa stabilisation, extérieure au processus lui-même, pourra être faite. C'est exactement ce que nous faisons dans nos simulations, en capturant des données en cours de processus pour afficher les courbes et autres informations.

En poussant la métaphore des nuées d'oiseaux, l'apparition de la membrane consiste en la formation de la nuée comme une entité à part entière. Une fois cette dernière opérationnelle les oiseaux peuvent s'y déplacer continuement sans la

10 Je ne peux m'empêcher de faire référence ici à la cosmologie, dont les avancées récentes nous indiquent avec certitude que, peu de temps après le Big Bang, il y eut ce que l'on appelle l'inflation. Au cours de cette inflation l'univers a connu une expansion prodigieuse, accompagnée d'une baisse de température tout aussi prodigieuse. Elle s'est ponctuée assez rapidement par un ralentissement très net de l'expansion. C'est à ce prix et uniquement, que la matière a pu exister, et donc les galaxies, puis la vie telle que nous la connaissons aujourd'hui. Nous savons d'ailleurs, qu'à l'heure actuelle, l'expansion forte a repris. Les lois de la physique ayant émergées grâce à ce ralentissement, l'univers peut fonctionner malgré cette nouvelle expansion forte. C'est donc un ralentissement, une forme de lenteur, qui a permis l'apparition de ce que nous sommes…

remettre en cause tout en jouant sur l'élasticité de ses paramètres internes (densité, vitesse et direction de déplacement, représentation topologique globale,…).
Dans notre processus de résolution, une fois la membrane constituée, la nuée de serveurs devient robuste (élasticité de ses propres paramètres) à des variations locales comme l'évolution de la demande des clients, le nombre de clients, voire leur position. Si nous avions continué nos études, nous aurions évoqué cette robustesse dans le processus de différenciation pour produire et reconnaître des instances de modèles.

Le chapitre III a mis en valeur un certain nombre de propriétés fondamendales utiles pour la résolution du problème C_nS_k de partage de ressources:

- nous savons (dans la partie III-A2 du chapitre III) que le coût global (CG_k) de la solution trouvée en fonction de la profondeur 'k' de la résolution ('k' correspond au nombre de serveurs de la nuée) décroît avec 'k' croissant, pour valoir 0 lorsque k=n, où 'n' est le nombre des serveurs qui a atteint celui des clients du système. Suivant notre définition de la fonction de coût, qui ne prend en compte que la distance du client au serveur associé, il est évident que si l'on place un serveur sur chaque client, le coût global résultant est nul. Il est tout autant évident que cette solution extrême n'a aucun intérêt, cela reviendrait à superposer les notions de client et de serveur. Je profite de cette remarque pour introduire ici la possibilité d'utiliser dans MVB de multiples fonctions de coût (CG) dont les caractéristiques propres pourraient apporter des enrichissements et des complexifications potentielles au processus de résolution. Cela pourrait faire l'objet de nombreuses études intéressantes. Pour revenir à MVB nous avons imaginé une exploration en profondeur qui démarre avec un seul serveur dans la «nuée» (k=1), résolution du premier problème, puis avec deux serveurs (k=2), deuxième problème, et ainsi de suite jusqu'à 'n' serveurs (k=n), n^e problème. En effet nous considérons que nous faisons face à un nouveau problème par niveau de profondeur atteint;
- nous savons également (toujours dans la partie III-A2 du chapitre III) que la complexité de la recherche de l'optimum est une courbe en cloche, symétrique possédant soit un ou deux points d'inflexion, suivant la parité ou non du nombre de points sur la grille (D^2).

Le recours au diagramme de Voronoï, dans la partie III-D2 du chapitre III, pour la représentation des régions du partitionnement lié à toute solution, nous montre qu'à chaque nouveau niveau de profondeur (ajout d'un serveur en passant de 'k' à k+1) le système doit se «débrouiller» pour faire en sorte qu'au moins l'un des clients existant choisisse ce nouveau serveur. Cela peut se faire directement au moment du choix de la position du nouveau serveur qui ira vers l'un des serveurs existant le plus chargé en clients (c'est l'option que nous avons choisi dans MVB), soit assez rapidement par le jeu de l'algorithme C/E. Notre expérience dans MVB est qu'un attracteur inintéressant se produit assez facilement si la position initiale du nouveau

serveur n'est pas pertinente. En fait l'algorithme C/E n'arrive pas, dans ce cas, à déplacer ce nouveau serveur vers l'un des serveurs le plus chargé, le conduisant très vite à une famine de client (attracteur). Cette situation est inutile car le système resterait au niveau 'k', même si le nombre de serveurs à crû d'une unité (niveau k+1). Grâce à notre heuristique de placement initial du nouveau venu, le système fonctionne très bien.

Dans le cadre des nuées d'oiseaux, après la constitution de la membrane, donc de la nuée, il s'agit d'utiliser l'espace interne à cette membrane (volume de la nuée) pour absorber les arrivées et les départs des oiseaux de la nuée. La gestion de l'espace dans la nuée, via le comportement individuel des oiseaux (l'équivalent des règles de Reynolds) conduit à garder une densité relativement stable dans la nuée, s'autorisant d'éventuelles augmentations ou diminutions du volume global.
Sans en avoir apporté la preuve (c'est une étude intéressante à faire…), mais juste constaté dans les faits au travers des nombreuses expérimentations réalisées, nous faisons l'hypothèse que si la résolution du problème d'allocation de ressources conduit à des solutions optimales et consécutives aux premiers niveaux traités (k=1, k=2, k=3,…) alors la suite de la résolution pourra atteindre les solutions optimales aux niveaux de profondeur supérieurs. Entendons-nous bien, le terme «pourra» joue un rôle très important dans notre hypothèse. Dans MVB l'exploration fonctionne par itérations successives (dont le nombre est soit configurable -solution retenue par manque de temps- soit calculable automatiquement) pour une profondeur 'k' donnée, puis descend au niveau de profondeur supérieur (k+1). Notre hypothèse résolutoire est que la solution optimale pour une profondeur k+1 donnée est accessible si et seulement si les solutions optimales des profondeurs plus faibles l'ont été à chaque niveau. Charge au système de définir le nombre d'itérations nécessaires pour cela; ce qui n'est pas une question triviale. Dans MVB nous n'avons pas apporté de réponses satisfaisantes à cette question. Le système traverse des solutions sans savoir a priori leur valeur par rapport à l'optimale ('=' ou '>'). Tant que l'exploration n'est pas terminée (d'où le problème de l'arrêt) on peut imaginer trouver une meilleure solution. Nous fondons notre hypothèse sur nos expérimentations que nous avons pu comparer aux valeurs optimales calculées de façon exhaustive. Ne disposant pas de moyens de calculs énormes, nous n'avons pu vérifier nos expérimentations que sur des exemples relativement modestes (cf. la partie IV-D du chapitre IV). La démonstration pourraît être faite par récurrence, puisque nous savons que c'est vrai dans les premiers niveaux calculables, et que ça l'est également pour les derniers niveaux (en particulier pour k=n).

Pour conclure cette brève introduction à notre approche autopoïétique de la résolution de problèmes, nous allons introduire ce que cette approche nous inspire dans le cadre de la représentation des connaissances.
Si nous supposons qu'un modèle (en représentation des connaissances) est l'expression de l'émergence d'entités individuelles en interaction au sein d'un processus de

différenciation, consommateur de ressources partagées et réutilisable à un niveau de représentation supérieur, en un mot un processus autopoïétique, alors notre approche (MVB) peut être mise en œuvre sur des modèles simples, comme par exemple celui des lettres de l'alphabet latin. Chaque lettre devient un modèle, dès lors que des interactions aient été faites entre leurs propres occurrences, et, dans le même temps, que d'autres interactions aient été faites entre leurs occurrences et celles des différents modèles. C'est l'instanciation dans la différenciation.
Le recours à MVB dans ce cas s'inspire également de l'idée que la représentation de connaissances doit reposer sur un principe d'optimisation des ressources nécessaires.

Vers la représentation de connaissances : l'alphabet latin

L'idée, ici, est de comprendre, ou en tous cas d'explorer, le principe d'instanciation de modèles (une lettre de l'alphabet équivaut à un modèle) dans un processus de différenciation en jeu dans la représentation des connaissances. Pourquoi peut-on reconnaître toutes les instances possibles du modèle 'X' parmi les instances possibles des autres modèles (comme par exemple 'Y') ? En quoi l'existence de ces différents modèles (l'alphabet) permet ces reconnaissances ? Enfin qu'est-ce qui lie ces différents modèles entre eux pour pouvoir les combiner sans ambiguïté «excessive» vers des modèles de niveaux supérieurs (mot…) ?

Nos simulations (MVB) ont permis de mettre en évidence la dynamique de résolution du problème de partage de ressources (C_nS_k) en se basant sur une recherche de multi-points optimaux à différents niveaux 'k' de représentation. Si k=1 cela signifie que l'on cherche le barycentre optimal qui représente le motif étudié (une lettre dans notre cas). Or il est simple de voir que pour des couples de lettres comme (X/Y) ou (O/D), cette seule représentation (k=1) ne permettra pas de faire une différence significative entre des instances de ces deux modèles qui sont assez proches. En revanche l'exploration en profondeur (k=2,3,4,5,…) peut permettre cette différenciation d'instances en fonction des caractéristiques réciproques de ces couples de modèles. C'est exactement ce que propose MVB. Nous savons que la limite de la profondeur 'k' est le nombre de points que nous utilisons pour représenter l'instance étudiée.
En effet si 'k' vaut cette valeur (nombre total de points représentant l'instance) alors la représentation de l'instance sera confondue avec l'instance elle-même.
Dans la mesure où deux instances de modèles distincts sont différentes, il ne sera pas nécessaire de descendre à cette profondeur limite. Il existe une valeur pour 'k' qui différencie ces deux instances entre elles.

Nous pensons que l'étude des dynamiques croisées des représentations (C_nS_k) des différents modèles (lettres de l'alphabet) enferment des informations susceptibles d'apporter des réponses à nos questions préalablement introduites.

Les résolutions MVB fonctionnent par exploration de l'espace des solutions (états) sans aucune forme d'apprentissage. On peut parler d'écoulement dans une topologie définie par des gradients, des frontières,… à l'image de l'eau et de l'érosion qu'elle produit sur la terre sous l'effet de la gravitation.
Nous avions abordé la problématique de ce que nous avions appelé la *dérive des connaissances*. Ce principe, basé sur une gravitation informationnelle et interactionnelle entre les connaissances, était, à l'origine, pensé pour oublier/restructurer les connaissances d'une base de connaissances. Il s'agissait d'intégrer dans la base de connaissances, les interactions que les connaissances entretiennent entre elles, via une utilisation externe de cette base. En gros était modélisé l'oubli des connaissances en se basant de façon concomitante sur l'écoulement du temps et les interactions que subissaient ces connaissances entre elles, via un système d'excitation externe à la base.

Si l'on projette la *dérive des connaissances* dans MVB, on enrichit l'engrammage des connaissances, représentant un premier niveau de singularité de ces dernières, avec un modèle relationnel, celui des interactions, constitutif de niveaux supérieurs de singularité.

L'apport de la *dérive des connaissances* dans MVB devrait aussi nous permettre de construire sur cette base une nouvelle version qui intègrerait une forme d'apprentissage au cours des explorations. Il s'agirait alors de capitaliser les connaissances acquises lors des différentes explorations pour construire une représentation exploitable et partagée de l'espace des états.
Le passage par cette nouvelle étape est indispensable pour capitaliser sur l'expérience acquise pendant les explorations, et, non des moindres, pour aller vers de la représentation de connaissances, but ultime de nos travaux.

Enfin une extension (MVB^{++}) à l'actuel moteur (MVB) pourra être envisagée afin de construire les modèles, non plus sur la base d'une représentation canonique, comme c'est le cas dans MVB, mais sur celle d'instances/occurrences des dits modèles. Pour cela il suffirait de déplacer les clients entre chaque résolution conformément aux spécificités topologiques de ces instances.

Les nuées d'oiseaux nous apprennent que de la connaissance peut émerger spontanément, à un niveau supérieur, sous l'effet d'entités en interactions partageant de l'information, tout en restant ce qu'elles sont.

Pour des raisons personnelles nous avons décidé de ne pas poursuivre ces explorations toutes plus passionnantes les unes que les autres !

Le chapitre V avait pour but de présenter les principaux résultats et principes de la *dérive des connaissances* et de les appliquer à MVB pour produire MVB^{++} et partir directement des occurrences de modèles.
Le chapitre VI devait nous permettre d'appliquer MVB^{++}

aux lettres de l'alphabet latin et voir en quoi leur structures profondes (dynamique de représentation multi-niveaux) favoriseraient la construction/manipulation de connaissances d'un niveau supérieur (langage)…

Vers un placement crypté de nuées de serveurs

Arrêtons nous un instant sur un élément intéressant, soulevé par ce chapitre IV, à propos des nuées de serveurs. MVB nous permet de résoudre un problème complexe (en complexité algorithmique) en ayant recours à une résolution exploratoire, entièrement distribuée par des entités autonomes (fonctionnant avec le «pseudo-parallèlisme» du au moteur *oRis*), qui sont regroupées sous forme de nuées. Ces entités indépendantes et autonomes sont capables de se synchroniser, à l'image des nuées d'oiseaux, en s'appuyant sur des données «partagées».

Revenons en détails sur ces principes pour mieux comprendre la transposition de ces résultats vers un monde réellement distribué (Internet) où le parallèlisme est total, et qui aboutit au paradoxe de brider le monde réel pour optimiser son fonctionnement.

On peut voir le comportement des serveurs dans MVB comme un vaste système crypté dont le secret à garder est la position des membres de la nuée dans des moments critiques (par exemple un repli sur des sites sûrs afin d'échanger, produire ou transformer des données sensibles, ou un positionnement global optimal sur le réseau à des fins d'optimisation de l'usage des ressources,…).

Contrairement à la cryptologie classique où l'information échangée est codée avec un système de clés et des propriétés mathématiques élaborées, ici tout se fonde sur la notion de déplacement «pseudo-simultané» et le recours à une propriété mathématique simple: l'addition est surjective dans l'espace des réels.
Cette propriété, ajoutée à la mobilité et l'autonomie des serveurs, rend très difficile de connaître les valeurs locales des coûts de la solution globale optimale ou choisie collectivement sur d'autres critères. Chaque valeur, locale à un client, est le résultat de sa position avec celle du serveur en jeu dans l'interaction.

Or la sommation des valeurs locales (plusieurs clients et leur serveur) rend impossible une déduction de la position correspondante du serveur vis-à-vis de ses clients. C'est cette valeur qui est échangée, donc interceptible, sur le réseau entre les différents serveurs en jeu dans une résolution collective. Chaque serveur mémorise, sans aucun échange externe, chacune de ses positions correspondant aux valeurs échangées.

Cette première expertise nous ouvre la voie à une réflexion sur l'étude d'un placement crypté des nuées de serveurs dans un environnement «réel» de type Internet. Cette réflexion vient compléter un travail engagé dans les deux thèses, que j'ai dirigées, et qui ont été soutenues en

2010 et 2012 par Hugo Pommier et Benoît Romito. Ce sera, pour nous, l'occasion de revenir sur les propriétés d'un simulateur multi-agents comme *oRis*, et de l'application des résultats obtenus en simulation dans un vrai univers réparti, où le parallèlisme est total et non simulé.
Ce sera d'ailleurs une très bonne transition avec le chapitre VIII où nous faisons l'éloge de l'épaisseur du temps et de la lenteur, dans le domaine artistique de l'encre.

Le vrai parallèlisme augmente la rapidité d'exécution de tâches, mais il ferme la porte à l'exploration de certains états/solutions pourtant disponibles dans la nature et qui pourraient être ceux qui nous intéressent.

La production artistique dans le medium de l'encre, telle qu'elle apparaît dans le chapitre VIII, n'est qu'une concrétisation tangible de ce principe que l'on retrouve dans la résolution de type C_nS_k.

Revenons sur le travail présenté tout au long de ce chapitre IV. Une solution au problème (C_nS_k) dans MVB, consiste à proposer un partitionnement des clients avec un serveur par partition et le meilleur gain/coût global associé. Cela implique de trouver les positions globales optimales des serveurs dans leur partition (région) respective. La résolution est «collective» car le meilleur (optimum) partitionnement global n'est pas nécessairement la somme des meilleurs partitions individuelles. En optimisant une partition (placement du serveur vis-à-vis des clients concernés) on peut dégrader la solution globale.

L'exploration de l'espace des solutions dans MVB, fonctionne sur la base d'un cycle à trois phases exploratoires principales qui sont l'algorithme de contraction/expansion (C/E), les descentes de gradient (G) et la marche aléatoire (A) :

- la première phase (cœur de l'exploration) se base sur des heuristiques qui permettent de mettre le système dans une zone favorable de l'espace des solutions. On entend par favorable une zone de l'espace des solutions où les solutions optimales devraient s'y trouver. On peut faire le parallèle avec une mise au point grossière sur un microscope. Tant que le système trouve de nouveaux états (nombre et positionnement des serveurs de la nuée) il reste dans cette phase, le but étant de trouver le plus d'états possibles pour espérer traverser l'optimum recherché ;
- la deuxième phase utilise la topologie de l'espace des solutions dans lequel les solutions sous-optimales se retrouvent au fond de puits, d'où le recours à des descentes de gradient. Pour poursuivre la métaphore il s'agirait alors de la vis micrométrique du microscope, autorisant un réglage plus fin ;
- enfin la troisième phase permet de sortir de ces solutions sous-optimales de l'espace des solutions en utilisant ce que l'on pourrait décrire comme un marcheur aléatoire. Les déplacements aléatoires dans l'espace des solutions restent cantonnées au partitionnement en cours, afin de sortir d'un minimum local en passant d'un état à un autre le plus continûment possible (notion

de proximité topologique dans l'espace des solutions/états). L'idée est de ne pas passer à côté d'un état intéressant.

Le système est asynchrone dans la mesure où chaque serveur est autonome, mais pas complètement indépendant des autres serveurs impliqués dans la résolution «collective». Il existe des interactions indirectes entre eux, qui passent par des requêtes résiduelles. Ce sont des requêtes en échec sur un serveur, qui poursuivent leur recherche d'un autre serveur pouvant satisfaire leur demande. À certains moments précis du cycle, les serveurs se synchronisent, à l'image des oiseaux dans les nuées.
La nuée de serveurs se constitue en fonctionnant, et ce de façon incrémentale. Chaque nouveau serveur, impliqué dans la résolution, intègre la nuée et met à jour des liens avec ses proches voisins.
Cela identifie un changement d'état du système. Dès lors une mise à jour des liens de proximité entre les serveurs voisins se propage rapidement afin d'intégrer ce nouvel arrivant auprès de tous les serveurs existants dans la nuée. Le nouvel arrivant se retrouve très vite (par percolation) reconnu de tous.

Comme nous l'avons déjà évoqué, les requêtes résiduelles créent des interactions indirectes entre les serveurs. Bien qu'autonomes l'exploration conduit très vite tous les serveurs à se connaître. Cela permet de mettre en place des processus de coopéreation entre eux, qui vont reposer sur la notion de «données partagées».
C'est l'équivalent des segments de mémoire partagée dans *UNIX/Linux*, mais en réparti à travers les réseaux. Lorsqu'une «donnée partagée», par tous les membres de la nuée, est modifiée par l'un des membres de la nuée, sa nouvelle valeur est immédiatement échangée en temps réel avec tous les autres membres de la nuée.

Cette propriété est simple à faire sur *oRis*, puisqu'à chaque top logique du moteur d'exécution, une seule instance des serveurs a la main. Il suffit, pour celle qui a provoqué la modification, de la répercuter, à l'intérieur du top logique courant, aux autres serveurs dont elle connaît l'identité. Toute incohérence sera très vite corrigée (aux tops logiques suivants) en évitant les conflits et les pertes de données récurrentes. Nous allons voir plus loin qu'il en est tout autrement dans un environnement réellement réparti comme Internet.

La première des «données partagées», que l'on pourrait appeler la méta donnée partagée, est donc la liste des noms (chaîne de caractères) des membres de la nuée. Cette liste est incrémentale et mise à jour à chaque changement d'état (incluant l'arrivée dans la nuée des nouveaux serveurs). Dans MVB nous ne gérons pas les sorties de serveurs de la nuée.

Il faudrait le faire lorsque, par exemple, un serveur devient inopérant. Il n'y a pas de difficulté conceptuelle à cela, mais nous n'en avons pas eu l'usage dans nos résolutions.

Très rapidement cette liste devient stable, car exhaustive, grâce à une percolation à deux niveaux (n+2) de profondeur des interactions directes via les requêtes résiduelles. En clair chaque serveur communique à ceux qu'il voit sa liste des serveurs de la nuée à laquelle il appartient (niveau n+1) et chacun de ces serveurs vus en fera de même (niveau n+2). Une fois cette liste «constituée» on peut définir d'autres «données partagées». Cela sera utile, par exemple, pour gérer la synchronisation de tous les serveurs de la nuée entre eux sur le mode interne RESTRUCT. Ce mode correspond, pour chaque serveur qui l'utilise, à son passage dans l'algorithme d'exploration contraction/expansion (C/E), première phase de l'exploration.

Nous avons vu précédemment que, dans le graphe des états internes des serveurs (cf. la figure *Fig-IV-E2-118*), il existe des points de synchronisation forte (l'équivalent de l'opération 'Z' dans les sémaphores d'*UNIX/Linux*) comme par exemple le mode interne -5 qui assure le passage en mode GRADIENT2, pour le serveur courant, uniquement si tous les autres serveurs arrivent au mode -5. Ce point de synchronisation permet d'attendre que tous les serveurs soient à l'équilibre, c.-à-d. que l'algorithme d'exploration C/E ne donne plus de nouveaux résultats (solutions). Il suffit que n'importe quel autre serveur repasse en mode RESTRUCT (exploration active de l'espace des solutions via l'algorithme C/E) pour que tous les serveurs en attente en mode -5 repassent également en mode RESTRUCT, relançant une nouvelle exploration complète.

On retrouve ce besoin de synchronisation à différents stades du graphe des états internes des serveurs.

C'est aussi le cas des modes -1 et -4. Le premier synchronise la nuée après les descentes de gradient locales, afin que tous les serveurs se calent en même temps (mode -2) sur la meilleure solution traversée.
Le second synchronise la nuée pour décider soit d'arrêter l'exploration, soit de repartir dans un nouveau cycle en restant au niveau 'k' (nombre de serveurs de la nuée), soit encore d'ajouter un élément à la nuée (k+1) pour entamer de nouveaux cycles d'exploration.

Nous avons décrit en détails tous ces mécanismes dans le graphe d'états de MVB. Le contenu de la variable locale décrivant l'état interne de chaque serveur peut devenir identique dans des phases de synchronisation de l'exploration, sans pour autant devenir une «donnée partagée».

Pour rappel chaque serveur dispose des «données partagées» suivantes :
- la liste des serveurs de la nuée courante;
- le vecteur des états (solutions), traversés par le système lors de l'exploration, et stockés sous la forme de réels;
- le nombre de serveurs en attente de la phase finale autopoïétique;
- le nombre de serveurs en attente en mode EQUILIBRE;
- le nombre de serveurs en attente en mode -4;
- les dernières meilleures valeurs de CG pour les états

parcourus ;
- le nombre d'explorations faites par la simulation en cours ;
- l'état perdu en début d'exploration.

L'autonomie des serveurs se matérialise par l'utilisation asynchrone du graphe des états internes. La coopération intervient grâce à ces variables partagées.

Autant le maintient de la liste des serveurs de la nuée ne pose aucun problème dans *oRis*, autant cela devient un sérieux problème dans un environnement réellement réparti, où les serveurs peuvent se déplacer exactement au même moment à l'échelle du réseau : ce qui est le cas d'Internet. Des solutions techniques existent et ont été implantées avec succès. C'est le cas du système *MOOREA*[11] qui proposait une mobilité forte des agents dans les réseaux.
Dans ce système, lors d'une demande de mobilité, l'état de l'agent était sauvegardé (pile d'exécution et données), puis restauré après déplacement, et ce quelque soit l'instant dans l'activité de l'agent. En outre un système d'agents fantômes était mis en place pour garantir la continuité des requêtes en cours lors des déplacements des agents originels. Les requêtes envoyées, avant la mobilité des agents, pouvaient arriver sur des agents fantômes qui redirigeaient ces requêtes vers les agents originels, le temps de mettre à jour les informations de localisation des agents mobiles. Aucune requête n'était perdue à cause de la mobilité des destinataires.

Cette notion d'agent fantôme peut aussi être implantée en utilisant la métaphore des phéromones pour les fourmis qui joueraient ce rôle. Chaque serveur peut laisser des phéromones (à durée de vie limitée) dans le réseau lors de ses déplacements. Cela implique qu'une requête résiduelle identifie une cible via son identifiant et sa dernière position connue. En arrivant sur cette dernière position connue, si le phéromone existe encore (structure dissipative) la requête obtiendra la position suivante. Plusieurs stratégies de mise à jour des phéromones peuvent être étudiées afin d'optimiser la recherche du serveur cible.

Si le phéromone visé n'existe plus quand la requête résiduelle arrive sur zone, cela ne remet pas en cause le fonctionnement de l'exploration. En effet les requêtes étant furtives, cela signifie que dans leur fonctionnement habituel elles se détourneront vers un nouveau serveur plus proche que celui visé initialement. Donc la perte de la piste d'un serveur en mouvement revient à un changement de cible.
L'évaporation des phéromones conduit à leur disparition, évitant ainsi de conserver des entités inutiles dans le réseau. Le réglage de la durée de vie de ces entités devra être étudié et probablement auto-configuré en fonction de la dynamique exploratoire de la résolution en cours.

11 Dillenseger, B., Tagant, A.-M. et Hazard, L. (2002). « Programming and executing telecommunication service logic with moorea reactive modile agents ». In the 4th International Workshop Mobile Agents for Telecommunication Applications, MATA 2002, p. 48-57. Springer.

Comme nous avons pu l'observer lors des différentes simulations, l'exploration est un processus qui tend progressivement à neutraliser le déplacement des serveurs vers la solution retenue. Naturellement l'utilité et l'existence des phéromones devraient diminuer plus on avance dans la résolution.

La deuxième «donnée partagée», stratégique pour l'exploration, est la valeur du coût global (CG) de la solution en cours. Cette valeur implique le résultat local calculé dans chaque région définie par chaque serveur. La sommation de ces valeurs locales conditionne le comportement individuel de chaque serveur.
Plus précisément, à chaque changement d'état (solution globale) détecté localement par l'un des serveurs de la nuée (le premier qui constate un changement de configuration de sa région, c.-à-d. un changement dans la liste des clients de sa région), ce dernier demande à ses collègues de lui fournir le coût local (CG_R) de sa région avant le changement. Cela lui permet de construire le CG global de la solution qui vient d'être quittée. En retour il transmet à ses collègues cette nouvelle valeur («donnée partagée»). C'est de cette façon, sans échange supplémentaire, que chaque serveur met à jour sa liste des états traversés («donnée partagée»).
La «donnée partagée» CGetat permet à chaque serveur de connaître le meilleur état rencontré jusqu'alors. À charge de chaque élément de la nuée d'y associer des informations (peu nombreuses) comme sa position dans cet état, la liste de ses clients définissant la région correspondante (,…), qui ne seront jamais échangées sur le réseau.

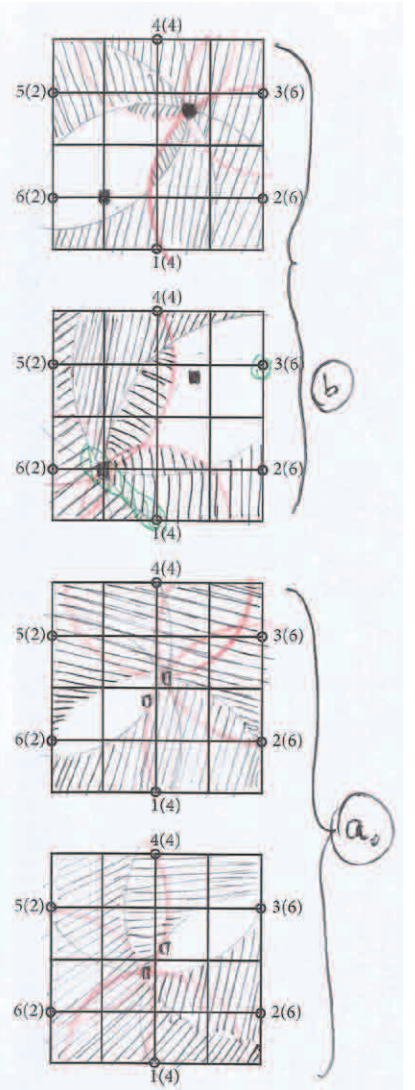


Fig-IV-E3-157

Éloge de la lenteur

La transposition de MVB à un réseaux distribué et ouvert comme Internet pose la question d'un paradoxe, celui de la parallèlisation des actions.

MVB est basé conceptuellement sur l'ordonnanceur logique et équitable d'*oRis* (cf. la partie IV-C du chapitre IV). Cet outil a été réalisé dans la mouvance des systèmes multi-agents, où l'idée était de reproduire sur machine centralisée, le comportement d'entités dans la nature, que ce soit des phénomènes biologiques (par exemple la modélisation de l'écoulement du sang dans les veines), physiques (par exemple l'écoulement de grains de sable dans un tas) ou encore sociaux (par exemple les systèmes proies-prédateurs dans les écosystèmes animaliers). En clair comment reproduire du parallèlisme observé dans des systèmes naturels, sur des systèmes informatiques qui sont hautement centralisés.
Pour cela le moteur d'*oRis* utilise, comme d'autres outils du même type, la notion de top logique permettant le découpage en micro-tâches des tâches des entités sensées fonctionner en parallèle. En donnant la main successivement à chaque micro-tâche des différents acteurs de la simulation, et en jouant sur une très grande rapidité d'éxécution de ces dernières (puissance de calcul des ordinateurs), *oRis*

introduit un pseudo-parallèlisme comportemental. En effet si l'environnement d'un acteur a changé entre l'exécution de deux de ses micro-tâches successives, alors ce dernier pourra changer son comportement attaché à sa micro-tâche à venir. Autrement dit les acteurs deviennent très réactifs à leur environnement, et donc aux autres acteurs, approchant ainsi le parallèlisme de la vrai vie…
Malgrè tous ses efforts remarquables, *oRis* n'arrive pas à atteindre le vrai parallèlisme. Deux micro-tâches, aussi petites en durée d'exécution fussent-elles, ne seront jamais exécutées exactement au même instant. L'une sera toujours effectuée avant l'autre.
Et c'est précisément dans cette propriété de pseudo-parallèlisme que s'est caché la réussite de MVB.
Le paradoxe veut que si l'on tranpose MVB dans la vraie vie distribuée et ouverte, où il règne un parallèlisme absolu (deux micro-évènements peuvent être parfaitement concomitants) MVB se heurte à un problème simple à décrire, mais plus complexe à résoudre.

Nous avons vu dans MVB qu'à chaque déplacement d'un serveur on peut potentiellement changer d'état (solution globale au problème d'allocation de ressources C_nS_k) dans la mesure où les clients sont fixes et travaillent avec le serveur le plus proche.

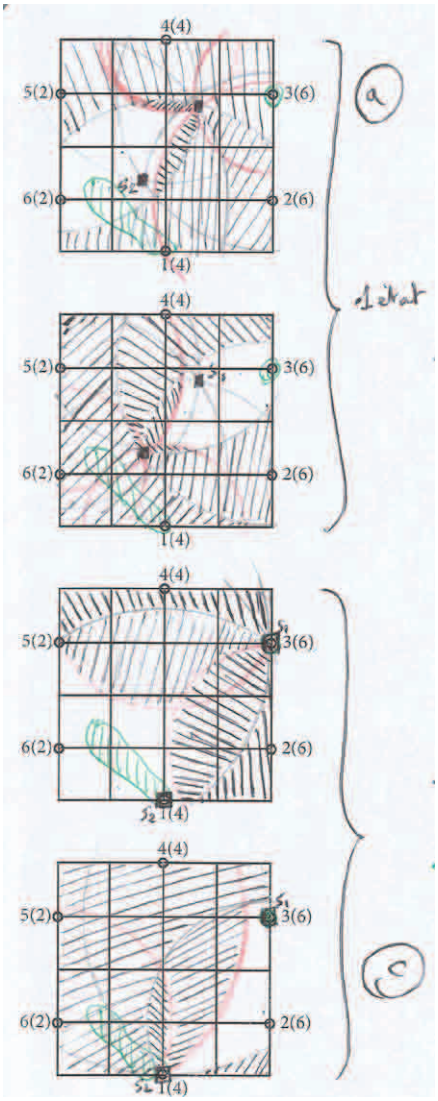


Fig-IV-E3-158

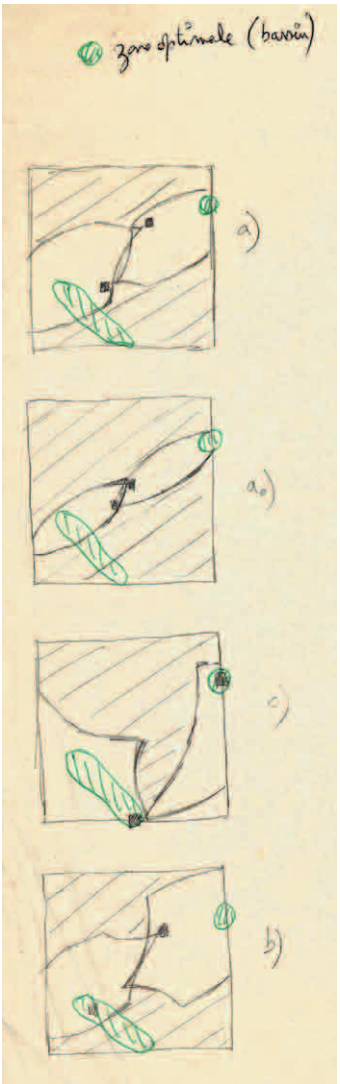


Fig-IV-E3-159

Pour illustrer la dynamique complexe des changements d'état/solution dans le processus exploratoire, prenons le cas simple où les deux serveurs S_1 et S_2 (petits carrés pleins sur les figures *Fig-E3-157/158)* se déplacent, au milieu des six clients n°1 à n°6 (petits ronds en périphérie de la grille) à la recherche de leur position optimale à l'intérieur d'un état/solution. Pour cela ils vont franchir dans l'ordre les étapes a_0), a), b) puis c).
Les figures *Fig-IV-E3-157/158/159* nous montrent (en hachuré vert) où sont les bassins correspondant aux zones optimales que les serveurs cherchent à atteindre dans l'état en cours. Les descentes de gradient (G), du cycle exploratoire C/E-G-A, permettent aux serveurs de se positionner dans ces bassins, obtenant ainsi un coût global minimal.
Sur les figures *Fig-E3-157/158* nous avons représenté deux fois la grille pour chaque étape des déplacements. En effet nous avons décomposé les déplacements des deux serveurs en supposant que l'un est fixe (S_1 sur la grille du haut et S_2 sur la grille du bas) et l'autre possiblement en

mouvement (S_2 sur la grille du haut et S_1 sur la grille du bas). Celle du haut montre la zone neutre (non hachurée) de déplacement du serveur S_2 (quart bas gauche), c.-à-d. sans changement d'état, alors que celle du bas montre la zone neutre du serveur S_1 (quart haut droit).
La figure *Fig-IV-E3-159* fait la synthèse en superposant les zones neutres (non hachurées) des deux serveurs. On voit bien, sur cette figure, qu'à chaque étape les deux serveurs disposent d'un chemin pour rejoindre leur bassin respectif. On voit également, à chaque étape, que les zones neutres évoluent. Plus le nombre de serveurs est important et plus l'évolution de ces zones neutres est complexe. Le déplacement d'un serveur peut couper la route de l'autre serveur vers son bassin, mais surtout peut provoquer un changement d'état/solution.
Concrètement, dans MVB, la résolution fonctionne (obtention de l'état optimal) car un seul serveur peut se déplacer à la fois (à l'intérieur d'un top logique du moteur *oRis*), ce qui permet de calculer l'état que l'on vient de quitter avant d'entrer réellement dans le nouvel état et partager des informations liées avec tous les autres serveurs de la nuée.
Dans l'exemple des figures *Fig-IV-E3-157/158/159* cela signifie que lorsque S_1 se déplace, S_2 est fixe et utilisera les nouvelles informations (sa zone neutre recalculée suite au déplacement de S_1) pour son propre déplacement à venir.
Sauf que dans un monde réparti et parallèle, nous pourrons avoir deux déplacements de serveurs parfaitement concomitants.

Sur la figure *Fig-IV-E3-160* nous avons une nuée constituée des quatre serveurs S_1, S_2, S_3 et S_4. Nous avons figuré en couleur le résultats des quatre déplacements de ces serveurs pour aller de la position collective initiale P_0 (en noir sur la figure, au temps 't') à la position collective finale P_4 (en orange sur la figure, au temps (t+4dt)) en passant respectivement par les positions collectives P_1 (en vert sur la figure, au temps (t+dt)), P_2 (en bleu sur la figure, au temps (t+2dt)) et P_3 (en rouge sur la figure, au temps (t+3dt)).

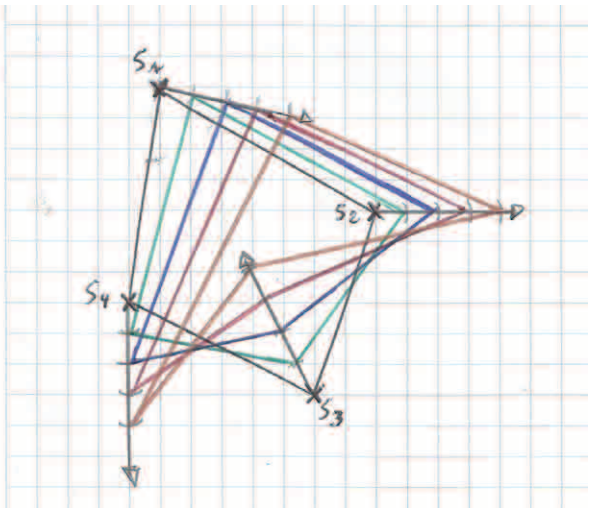


Fig-IV-E3-160

La figure *Fig-IV-E3-161* montre deux possibilités parmi plusieurs (en fonction de l'ordonnanceur d'*oRis*) pour passer séquentiellement par quatre étapes pour aller de P_0 à P_1, puisque chaque serveur fonctionne chacun son tour. Dans le premier cas (figure du haut) c'est S_1 qui se déplace en premier, puis S_2, suivi de S_3 et enfin S_4. Alors que dans le second cas nous avons S_3, puis S_2, suivi de S_4 et enfin S_1. Les couleurs permettent de voir à chaque étape les positions des différents serveurs entre eux.

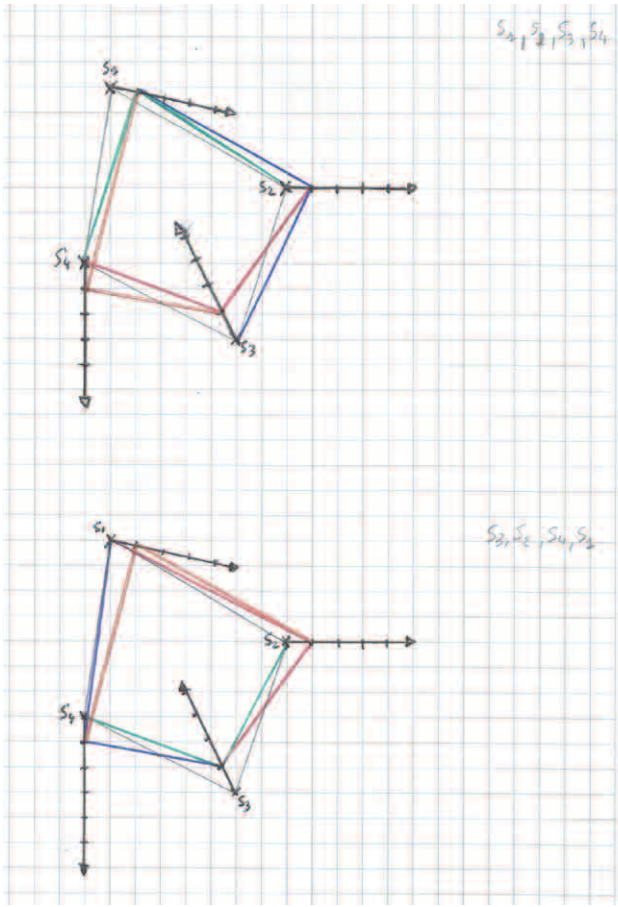


Fig-IV-E3-161

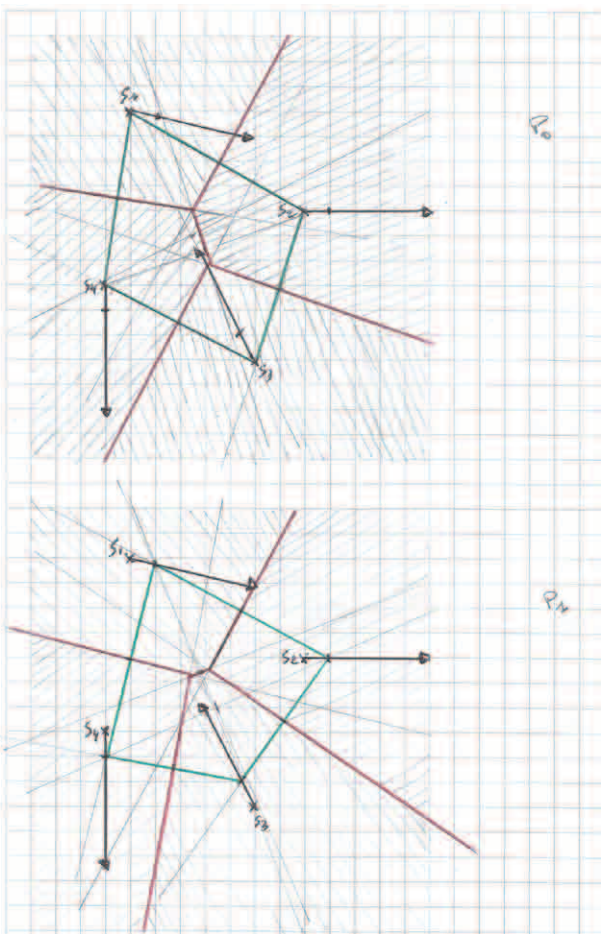


Fig-IV-E3-162

Comme le montre la figure *Fig-IV-E3-162*, où nous avons représenté les diagrammes de Voronoï respectifs des positions collectives P_0 et P_1, les partitions (délimitées par les traits rouges des régions propres à chaque serveur) évoluent assez nettement entre P_0 et P_1. C'est en parcourant le plus de régions possibles que le processus d'exploration aura des chances de rencontrer l'état optimal.
En changeant la topologie des régions nous changeons potentiellement la répartition des clients par région et donc nous rencontrons potentiellement de nouveaux états/solutions.

La figure *Fig-IV-E3-161* montre qu'*oRis* choisit déjà une solution parmi plusieurs pour aller de P_0 à P_1. À cause de cela tous les états potentiels ne sont pas systématiquement traversés. MVB trouve quand même la solution optimale grâce aux différentes itérations d'explorations menées à chaque niveau. En effet l'équité du moteur d'*oRis* permet de garantir que ce ne sera pas toujours la même micro-action qui sera choisie à chaque top logique. La probabilité de passer d'une position collective (P_0) à une autre (P_1) en utilisant le même ordre (ici avec quatre serveurs), à l'intérieur de la même exploration, voire dans deux explorations différentes, diminue avec le nombre de serveurs en jeu dans la résolution. Ce fonctionnement augmente donc les chances de passer par l'état recherché.

La situation est bien différente avec un parallèlisme absolu, comme Internet, puisque le déplacement, potentiellement parallèle, de plusieurs serveurs peut conduire à la disparition de positions collectives P_i. Cela diminue d'autant les chances de passer par les états recherchés.

Ce que l'on observe est qu'un parallèlisme absolu peut conduire à accélérer les déplacements des serveurs, mais en occultant une partie de l'espace des solutions, où pourraient se trouver celles que l'on cherche.

En fait le parallèlisme absolu rend l'exploration discontinue; on va plus vite, mais on voit moins de choses.

Nous avions largement et intuitivement considéré que l'exploration continue de l'espace des solutions était un gage de rencontre de l'optimal lors de l'exploration. L'idée d'une transposition de MVB dans Internet nous a fait prendre conscience de sa réelle importance.

Le paradoxe des actions parallèles nous conduit à considérer que la projection de VMB dans Internet nécessite, a minima, de reproduire le pseudo-parallèlisme d'*oRis* dans un univers où règne un parallèlisme absolu, mais trop rapide et surtout provoquant des discontinuités dans l'exploration de l'espace des solutions.

Pour cela il faut mettre en place des protocoles de communication entre les membres de la nuée de serveurs, comme par exemple ceux invoqués dans le calcul d'un temps unique partagé dans les systèmes distribués.

C'est tout le paradoxe d'avoir voulu copier la réalité que nous allons devoir transformer pour copier la copie et pouvoir bénéficier d'effets de bord positifs obtenus dans cette copie.

Le parallèlisme absolu ne modifie pas l'espace des solutions, mais il modifie la perception ou l'accès que nous en avons. Des zones, potentiellement importantes, peuvent être cachées, car devenues inaccessibles.

Or nous observons exactement le même phénomène avec la pratique artistique des encres que nous avons décrite au chapitre VIII. La lenteur du geste fait apparaître des paysages artistiques insoupçonnés.
C'est une révélation de choses existantes, mais cachées si le geste est trop rapide, donc inaccessibles.

Je ne pouvais trouver de meilleur lien autoréférent entre mon travail scientifique et mon approche artistique. L'un étant la concrétisation tangible de l'autre, la pratique au service de la théorie dans un mouvement circulaire et probablement hautement autopoïétique…

Ce mémoire clôt plusieurs décennies de recherches, d'errances et de doutes, malgré de nombreuses interrogations et pistes intéressantes encore à explorer. Le chapitre VIII fait le lien avec la suite à donner à mes activités dans ma nouvelle vie qui commence.

La publication de ce rapport me permet de passer définitivement la porte qui sépare ces deux mondes, pourtant largement porreux entre eux, et me dire que tout cela n'aura pas été vain.

Cela me permettra aussi de me séparer de tous ces feuillets et notes de tous poils que j'ai griffoné et qui m'ont accompagné pendant toutes ces années.

Avant de lâcher le crayon, je tiens une nouvelle fois à remercier tous ceux et toutes celles qui m'ont accompagné et fait confiance, et je sais qu'ils sont nombreux !

L'Estaca, Bricqueville-sur-Mer,
le vendredi 23 juin 2023.

LA *DÉRIVE DES CONNAISSANCES* : UNE APPROCHE COSMOLOGIQUE…

V

La *dérive des connaissances* fut l'aboutissement de mes premières années de recherche. Après un historique sommaire de ces aventures, qui devaient transformer profondément et durablement mes activités professionnelles, je vais jeter les bases intuitives de ce qui pourrait être la suite naturelle de ces travaux. En réalité il s'agit davantage de changer le point de vue sur ce qui m'occupa pendant toutes ces années. Toutes ces idées, toutes ces intuitions, auxquelles j'ai cru profondément, peuvent être relues dans le contexte des dernières avancées en matière cosmologique. À plusieurs reprises dans ce document, j'ai fait allusion au recours à *la dérive des connaissances* pour faire un lien entre la résolution C_nS_k, telle que je l'ai présentée aux chapitres III et IV et une approche très générale de la représentation des connaissances (objectif initial du chapitre VI). En effet la *dérive des connaissances* peut apporter une dimension d'apprentissage dans la résolution C_nS_k, en particulier au niveau de son espace des états, propriété indispensable vers le chemin de la représentation des connaissances.

De manière beaucoup plus profonde, la *dérive des connaissances* peut s'expliquer par une dimension nouvelle dans la résolution C_nS_k, introduisant, entre autres choses, une force gravitationnelle des relations entre connaissances, ouvrant la voie à une description spatio-temporelle de l'espace des états des connaissances manipulées et produites par le système.

V-A1) Les fondements du modèle

La *dérive des connaissances* est le cœur de mon travail de doctorat, présenté en juillet 1992 à l'Université du Maine au Mans. J'ai repris l'essentiel de cette modélisation, basée sur une approche gravitationnelle de l'information, dans mon rapport d'Habilitation à Diriger des Recherches (HDR), soutenue en janvier 1998 à l'Université de Caen. J'ai extrait de ce rapport les trente-six pages qui résument cette modélisation, et que je présente in extenso ici.

À l'époque de mon doctorat, j'avais en tête l'envie de modéliser un oubli «pertinent» de l'information. C'est ce que je tentai, sous la direction et la bienveillance de Martial Vivet, alors professeur à l'Université du Maine.
À mi-distance de mes travaux, alors en poste au S.E.P.T. à Caen, je pris rendez-vous avec Bernard Victorri, alors professeur à l'Université de Caen, dont j'avais suivi un cours en DEA en auditeur libre. Après m'avoir écouté patiemment, Bernard Victorri me fit un certain nombre de remarques, qui pouvaient laisser entendre une forme de compassion envers moi et ma situation pas facile, car en dehors du parcours académique classique. Sans me décourager, je poursuivis mes travaux.
Arrivé en fin de rédaction, je le contactai à nouveau pour lui sousmettre le manuscrit afin qu'il en devienne l'un des rapporteurs. Ce qu'il accepta, contre toute attente.
Il l'accueillit avec une très grande surprise au point de saluer ce travail comme «remarquable»… Ce fut également le cas de Jean-Marc Fouet, second rapporteur, ainsi que pour les autres membres du jury, dont Paul Bourgine. Cela me vaudra les félicitations du jury, moi qui avait fait ce travail en solitaire.
Cela m'encouragea à poursuivre sans pour autant publier ces résultats. Je n'ai même jamais implanté ce modèle, qui l'aura été par un collègue d'Avignon, assisté par un étudiant, plusieurs années après la parution de ma thèse. Je l'appris d'ailleurs par hasard, plus tard, lors d'un congrès où je croisais le dit collègue et son étudiant.
Avant d'implanter mon modèle, je me suis mis à chercher un cadre applicatif qui pourrait le mettre en valeur. C'est alors que j'ai commencé à réaliser des simulations multi-agents dans le contexte du partage décentralisé de ressources, dans un environnement distribué. Six ans après mon doctorat, je mettais en forme mes premiers résultats dans mon rapport d'HDR. Ces derniers s'émancipaient du modèle, pour vivre leur propre destin, apparemment très éloigné de la *dérive des connaissances*.
À de nombreuses reprises, je me suis interrogé sur une possible convergence des deux directions empruntées jusqu'alors. J'ai relevé plusieurs pistes possibles, mais aucune ne m'apparaissait comme une évidence. Ce n'est que récemment, lorsque j'ai entrepris cette aventure («Testament de Recherche»), que l'unification des deux approches semblait prendre tout son sens.
Je vais en décrire les grandes lignes intuitives dans la seconde partie (V-A2) de ce chapitre. Comme indiqué précédemment, je ne vais pas explorer plus en avant ces nouvelles pistes; peut-être le seront-elles un jour?

Extraits de mon rapport d'Habilitation à Diriger des Recherches (HDR), p.59-95 :

II . Modélisation des Interactions : "Le Modèle Relationnel"

1 . L'espace relationnel : ses composants, ses dynamiques.

1.1 . Avant-propos

Le modèle relationnel, baptisé "dérive des connaissances" [bou 92] est issu de réflexions sur l'évolution des systèmes informatiques susceptibles de supporter la gestion et l'utilisation des systèmes d'information des entreprises. La multiplicité des ordinateurs et des réseaux les reliant augmente dans des proportions vertigineuses les échanges de données électroniques. Cela pose le problème du rapport entre la notion de connaissance et celle d'information. Dans ce contexte, nous partons de l'hypothèse selon laquelle pour construire de la connaissance, il faut savoir *oublier* de l'information. Autrement dit, la connaissance est obtenue en restructurant les informations échangées. Se pose alors le problème de la définition de critères pertinents pour oublier ou restructurer l'information manipulée afin d'obtenir de la connaissance.

Comme nous le disions dans le préambule, avec l'arrivée massive d'ordinateurs communicant à grande échelle (cf. Internet XX-ref [réf 19XX]) et de réseaux performants XX-ref [atm 19XX], nous sommes arrivés à un tournant dans l'histoire des systèmes éléctroniques de production et de gestion d'information. Ce constat s'appuie sur un certain nombre d'éléments comme :

- ✗ un accroissement des capacités de stockage des ordinateurs, à coût et encombrement (miniaturisation) diminués,
- ✗ un nombre croissant d'échanges et de duplication des informations,
- ✗ des informations manipulées de plus en plus nombreuses, complexes, composites (audio, vidéo, texte, image ...), volumineuses, réparties, partagées ...

De ceci découle au moins deux problèmes majeurs pour ce genre de systèmes, qui sont la saturation des ressources (réseaux et machines) et la "poubellisation" de l'information. Le premier problème entraîne des difficultés de fonctionnement ; c'est le cas par exemple de certains serveurs WWW ouverts aux Etats-Unis, pour lesquels il est très difficile d'entrer en communication l'après-midi. Le second problème est celui de la transformation par le media de la connaissance en information ; cette transformation opère en diluant la connaissance dans une information extrêmement volumineuse, redondante et très difficile à exploiter (perte de sens).

Nous pouvons même énoncer ce qui apparaît comme l'un des paradoxes de l'information électronique :

> *Propriété-1* : Pour accéder à tout l'univers informationnel, il faut développer l'informatique communicante via les réseaux.
>
> *Propriété-2* : Cette communication informatique génère des échanges, qui entraînent de la redondance par duplication de l'information (copie-coller ...) ; cette duplication appauvrit donc le sens des informations manipulées ; cette perte de sens pousse donc à accroître encore davantage les échanges. Cette boucle infernale produite par la mise en place du tandem "informatique et communication", génère des informations inutilisables.

Dans WWW on peut soit concerver un pointeur vers une information distante, soit rappatrier cette information sur la mémoire locale. Dans le premier cas, on récupère l'information uniquement quand on en a besoin sans encombrer ses ressources de stockage locales, mais en étant tributaire de l'encombrement des réseaux et machines distantes ; dans ce cas l'information est toujours à jour par rapport à la source qui la fait évoluer. Dans le deuxième cas l'accès performant à l'information est garanti puisqu'on maîtrise les ressources locales, par contre l'information peut ne plus être à jour puisque la source de cette information à pu faire évoluer cette information depuis le dernier accès. Des travaux XX-ref (cf. Luigi Lancieri et Stephane) regarde les notions de pertinence de l'information dans le temps.

En combinant les deux propriétés issues de ce tandem on peut énoncer la propriété résultante suivante (paradoxe) : "En accédant à tout (*propriété-1*), on n'accède plus à rien (*propriété-2*) !"

Il apparaît à travers ces constatations un certain nombre de questions à prendre en compte pour gérer correctement les systèmes informatiques de production/manipulation d'information :

- Comment gérer un grand volume d'information ?
- Comment gérer des informations évolutives, partageables et probablement réparties ?
- Comment définir des critères de pertinence contextuelle pour les informations manipulées ?
- Comment prévoir l'évolution de ces critères ?
- Comment juger de l'obsolescence irrémédiable de certaines informations ?
- Comment détecter une inadéquation entre la structuration d'une information et la façon dont elle est manipulée ?
- Comment concilier intérêts individuels et intérêts collectifs en matière de partage de ressources ?

Plusieurs approches existent aujourd'hui pour résoudre ces problèmes ; beaucoup de travaux XX-ref [][][] regardent comment naviguer ("browser") intelligemment dans cet univers informationnel soit en utilisant des techniques à base de systèmes multi-agents [][], soit en travaillant sur de l'indexation couplée à de la recherche sur le contenu [][] ; d'autres proposent des éléments d'infrastructure comme les annuaires de service (trader) [][] afin de faciliter les contacts entre les offres et les demandes ; d'autres travaillent davantage sur des techniques issues des travaux sur le langage naturel (analyse lexicale, grammaticale ...) [][], d'autres encore (activité en forte effervescence XX-ref (cf. Luigi Lancieri) travaille sur la notion de cache d'informations. Ces approches sont basées en général sur la sémantique (sens) véhiculée par l'information manipulée. En terme de client/serveur cela se traduit par la définition du ou des services que peuvent rendre les serveurs et le ou les services qu'attendent les clients. Ce langage commun permet entre autre de définir les conditions dans lesquelles devront être opérées les transactions ; conditions portant bien entendu sur les services eux-mêmes (qualité de service).

Le modèle relationnel considère une information comme une entité évolutive qui, au-delà du sens véhiculé par son contenu, possède certaines caractéristiques qui rendent comptent de l'évolution de ses interactions avec son milieu. Cela part de l'hypothèse selon laquelle toute entité individuelle (petite granularité) existe aussi et peut-être surtout à travers le rôle qu'elle joue au sein de collectivités[1]. Ce rôle se définit suivant plusieurs facettes dont celle des interactions qu'entretient cette entité avec les autres. Ces interactions et une certaine représentation de ces différents contextes qui les ont produites, deviennent alors naturellement une partie de ce qui définit ces entités. C'est une sorte de mémoire de l'entité dans sa perspective évolutioniste, c'est-à-dire qui intègre une vision du contexte dans lequel elle s'est développée. C'est à partir de cette "mémoire" relationnelle représentant les conditions d'évolutions des interactions des entités entre elles, que le modèle relationnel propose des critères de pertinence pour transformer de l'information en connaissance.

Le modèle relationnel est applicable aux systèmes qui fondent leur auto-adaptation sur les phénomènes stables (reproductible dans le temps et dans l'espace) qu'ils produisent ; cela exclue donc la prise en compte d'éventuels épi-phénomènes, qui sont considérés ici comme du bruit. Ces contraintes permettent alors au modèle relationnel d'être un modèle descriptif, car basé sur une auto-observation continue du fonctionnement réel du système. Il est aussi préventif dans la mesure où il permet la détection des changements de tendances dans les interactions. Enfin il sert de guide pour une auto-organisation collective du système conformément à ces changements. Cela signifie que l'utilisation individuelle du modèle permet une compréhension "collective" et non pas uniquement locale ou individuelle des phénomènes observés. Nous verrons que la notion de "collectif" est à géométrie variable et qu'elle coûtera en fonction de son étendue.

Principe d'observation

L'application du modèle d'interaction[2] repose sur des périodes d'observation pendant lesquelles le système observé fonctionne normalement. C'est-à-dire avec une organisation qui est issue d'une part du résultat des déci-

1. Maturana et Varela fondent leur théorie des êtres vivants sur la notion de machine autopoiétique. Pour eux " *... les êtres vivants sont des sortes de machines définies par leur organisation, qu'on peut expliquer comme on explique n'importe quelle organisation, c'est-à-dire en termes de relations entre composants et non en se basant sur les propriétés de ces composants ...*" XX-ref (La vie artificielle Heudin p162).
2. Nous parlons indifféremment dans le texte du modèle d'interactions ou du modèle relationnel.

sions de réorganisation liées aux périodes d'observation antérieures et d'autre part à l'évolution du système lui-même. Dans l'exemple du partage de ressources (cf. chapitre 4-III) le type de réorganisation possible suite aux observations antérieures est le déplacement d'un serveur vers des clients, alors que celui inhérent au système lui-même est par exemple l'ajout d'une machine ou encore le changement de comportement de clients. Nous verrons sur l'exemple présenté, que, suivant les heuristiques de réorganisation choisies, le système évolue vers des situations non nécessairement optimales vis-à-vis de critères à préciser, mais tout au moins acceptable toujours vis-à-vis de ces critères. Nous montrerons en particulier que suivant l'implication du modèle relationnel dans les stratégies de réorganisation choisies, on aura des résultats qualitativement différents.

Pendant la période d'observation courante on fait évoluer la configuration relationnelle conformément à l'évolution des interactions pendant cette période. En fin de période chaque composant du système examine l'adéquation entre la description relationnelle et la description opérationnelle qu'il a du système (lui-même plongé dans son contexte). En cas de divergence même partielle de ces descriptions, chaque composant met en oeuvre un processus décisionnel dont l'effet conduit à des modifications de la structure opérationnelle du composant lui-même et par effets de bord, plus généralement du système. Nous verrons dans l'exemple (chapitre 4-III) comment une action de réorganisation locale à un serveur, comme la duplication du serveur, peut conduire d'une part à modifier les interactions de ce serveur avec ses clients, mais aussi provoquer des modifications entre d'autres serveurs et leurs clients.

Le paragraphe suivant décrit les trois points de vue fondamentaux (local, collectif et global) qui constituent le cadre de la description relationnelle des interactions entre les entités. Ensuite nous abordons les notions de *flux excitatoire* et de *flux entropique* qui introduisent une dynamique d'évolution au niveau des valuations des liens relationnels entre les entités. Cette dynamique est paramétrable suivant les heuristiques qualitatives choisies pour le domaine applicatif visé. Nous abordons dans ce paragraphe comment elle intègre une modélisation mécanistique (attraction par masse relationnelle) et une modélisation topologique (forme collective). Enfin nous présentons dans une dernière partie, un mécanisme important dans la dynamique du modèle, puisqu'il est à la base de la détection de formes collectives stables et de la navigation dans ces formes collectives.

1.2 . Les trois points de vue fondamentaux

L'espace relationnel se décrit suivant trois points de vue qui se complètent et s'influencent mutuellement pour introduire les heuristiques comportementales (qualitative) choisies, dans le modèle numérique d'évolution des paramètres descriptifs. L'espace relationnel se compose d'un graphe non nécessairement connexe, où les noeuds représentant les informations en interaction, sont reliés entre eux par des liens fictifs. Les valeurs attachées à ces liens qantifient la nature des interactions entre les noeuds correspondants.

Le point de vue local : les noeuds

Un noeud est un représentant relationnel d'une entité réelle (processus/objet/serveur) dont on veut caractériser le fonctionnement par observation de ses interactions dans le milieu (système).

La masse relationnelle d'un noeud est une mesure *instantanée*[1] des interactions qu'entretient ce noeud avec les autres noeuds de la base. Cette mesure quantitative peut être utilisée pour comparer relationnellement des noeuds entre eux ; ces comparaisons donnerons des indications qualitatives sur le rôle relationnel de chaque noeud de la base. La variation de cette mesure pour un noeud K_i sera d'autant plus grande sur une période donnée que les interactions entre K_i et les autres noeuds seront nombreuses et intenses[2]. On parlera de la même façon de la masse relationnelle de l'ensemble de la *base* ou encore de la masse relationnelle d'une *forme relationnelle collective*. Dans ces cas il s'agit de paramètres dits de composition, puisqu'ils représentent l'ensemble des rôles relationnels individuels de chacun des composants de ces entités collectives. Pour une *forme relationnelle collective* donnée on peut définir sa masse relationnelle comme étant la combinaison des masses relationnelles de chacun des

1. Cf. XX-ref [bou 92] où l'on décrit que la masse d'un noeud au temps T ne dépend pas de sa valeur au temps T-ΔT
2. L'intensité indique simplement que suivant les domaines applicatifs visés, un lien relationnel ne traduira pas uniquement une fréquence d'interaction entre deux entités mais pourra intégrer des notions comme le volume des données échangées.

noeuds inclus dans cette forme. De la même façon la masse relationnelle de la *base* est une combinaison des masses relationnelles de chacunes des *formes relationnelles collectives* incluses dans cette base.

Nous verrons dans les calculs de flux excitatoire et entropique, comment certains rapports de masse entre les trois niveaux évoqués (noeud, forme collective et base) précisent les propriétés qualitatives que l'on cherche à faire jouer au modèle relationnel.

Les noeuds du graphe sont reliés par des liens qui traduisent un niveau d'interaction entre les éléments liés. Un noeud peut donc subir des interactions, quand il est sollicité par d'autres noeuds ; il peut également provoquer des interactions quand il sollicite d'autres noeuds.(1)

En reprennant la notion de noeud telle qu'elle a été définie dans XX-ref [bou 92], la masse relationnelle (M_{Ki}) d'un noeud K_i se compose de trois parties distinctes appelées MS_{Ki}, MR^-_{Ki} et MR^+_{Ki}. Avec :

$$M_{Ki,t} = MS_{Ki} + MR^+_{Ki,t} + MR^-_{Ki,t}$$

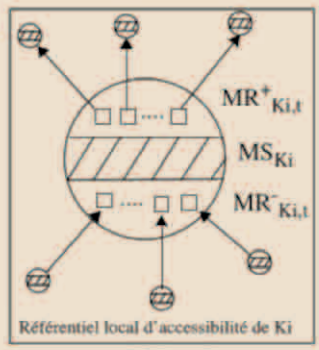


Schéma-1

- MS_{Ki} est la partie constante de la masse relationnelle de K_i ; elle correspond à la structure de K_i.
- $MR^+_{Ki,t}$ est la partie de la masse relationnelle de K_i issue des échanges provoqués par K_i vers d'autres noeuds du graphe.
- $MR^-_{Ki,t}$ est la partie de la masse relationnelle de K_i issue des échanges d'autres noeuds du graphe vers K_i.

La distinction de ces trois composantes et leur mode de calcul donne au modèle de propagation des perturbations aussi bien excitatoires qu'entropiques, un fonctionnement local ou encore asynchrone. Sans revenir en détail sur ces calculs qui sont décrits dans [bou 92] nous rappelons que MR^+_{Ki} est une fonction des MS des noeuds K_j vus par K_i, alors que MR^-_{Ki} est une fonction des MS des noeuds qui voient K_i. Ceci peut se schématiser par le dessin suivant ; au passage on voit sur ce dessin les topologies possibles d'assemblage des différents noeuds du graphe entre eux.

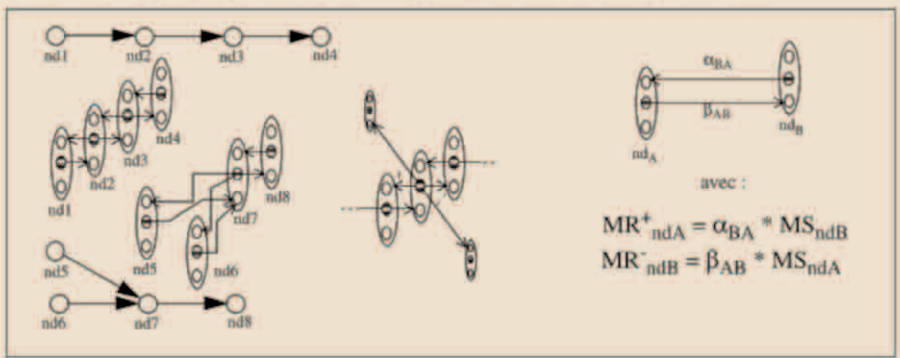


Schéma-2

Topologies qui sont baties sur la structure interne des noeuds et qui montrent que les liens d'interaction entre les noeuds se composent de liens internes asymétriques ($\alpha_{BA} \neq \beta_{AB}$ et $MS_{ndA} \neq MS_{ndB}$).

Ces définitions illustrent la notion de point de vue individuel ou local dans le système observé. On parlera de référentiel local d'accessibilité (cf. schéma-1) pour fixer ce point de vue individuel ; les calculs de masse et autres paramètres propres à un noeud seront effectués dans ce cadre. En effet chaque noeud du graphe de la base possè-

1. Quand un noeud K_i du graphe subit des interactions en provenance d'autres noeuds on dira que K_i *est vu* par ces derniers ; alors que pour les noeuds qui provoquent ces interactions avec K_i on dira qu'ils *voient* K_i.

de sa propre vision de la base au travers des liens directs qui le relient avec d'autres noeuds de la base. Cette vision locale présente dans les modes de calcul des masses relationnelles comme dans ceux d'autres paramètres relationnels, correspond à l'idée qu'un noeud donné ne doit pas intégrer les conséquences indirectes des perturbations/interactions dont il est la cause. Nous reviendrons sur cette propriété lorsque nous parlerons des flux excitatoires.

L'une des propriétés des systèmes observés, via le modèle relationnel proposé, est que ces éléments (noeuds) interagissent entre eux de façon concurrente[1]. Cela signifie que certains noeuds sont activés *simultanément* par plusieurs noeuds différents ; c'est le cas de plusieurs clients qui accèdent à un même serveur, dans le paradigme informatique du Client/Serveur. En fait le modèle ne travaillant pas en temps réel mais sur des périodes d'observation longues, la simultanéité signifie que sur une période donnée un noeud a été activé par plusieurs autres noeuds.

Cette propriété implique le modèle à travailler à un niveau collectif qui dépasse la vision individuelle de chaque noeud de la base. Pour prendre en compte ce phénomène et celui des propagations d'excitation[2] qui peuvent dépasser encore une fois la vision individuel des choses, nous introduisons dans le modèle la notion de forme relationnelle collective.

Le point de vue collectif : les formes relationnelles collectives

Cette notion de forme collective se distingue des deux autres dans la mesure où elle est dynamique. Tant qu'une information existe, elle reste un noeud pour le système relationnel ; il en est de même pour la base. Cela n'empêche pas de voir leur description relationnelle évoluer dans le temps. Par contre les formes relationnelles collectives vont émerger, disparaître ou se transformer (agrégation, séparation ...) au cours du temps.

Une *forme relationnelle collective* est un sous-graphe connexe de la base qui est activé sous l'impulsion d'excitations. Elle correspond à un regroupement d'entités fonctionnellement liées entre elles. Pour qu'une forme joue un rôle dans le modèle relationnel, il faut qu'elle soit homogène pendant toute la durée de son existance. L'homogénéité définit une propriété collective de la forme qui autorise des transformations *identiques* de chacun de ses éléments sans remettre en cause l'identité de cette dernière. Lorsque l'évolution des interactions entre les noeuds du sous-graphe qui compose une forme, le permet, cette propriété d'homogénéité garantit la conservation des formes à partir d'une structure invariante (propriété de stabilité). En d'autres termes, si tous les noeuds d'une forme évoluent de manière identique, suivant par exemple une augmentation proportionnelle des interactions, alors la forme est concervée en tant que telle. L'idée est que les formes représentent les entités collectives stables que l'on cherche soit à identifier, soit à faire émerger dans la base afin d'adapter la structuration de la base au bon fonctionnement de ces entités stables. Elles sont observables continuement dans le temps si elles évoluent de façon "homogène" ; lorsque des parties de ces formes évoluent différemment des autres parties restantes, il se produit alors une rupture d'homogénéité dont la détection permet la transformation de la forme étudiée par séparation ou autre mécanisme.

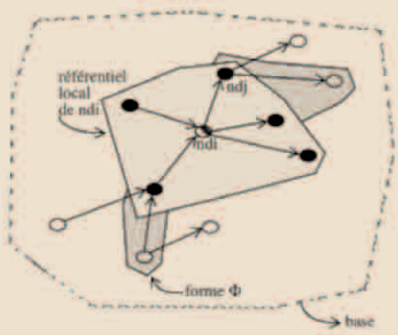


Schéma-3

1. C'est en partie pour cette raison que nous avons fait introduire dans le simulateur la technologie réactive XX-ref (cf. chapitre 5-II), issue des langages synchrone, grâce à laquelle nous pourrons gérons correctement des mécanismes d'accès et de traitement concurrents de l'information, comme ils existent dans les systèmes réels.
2. Un noeud active un noeud qui va activer à son tour et consécutivement un troisième noeud etc ...

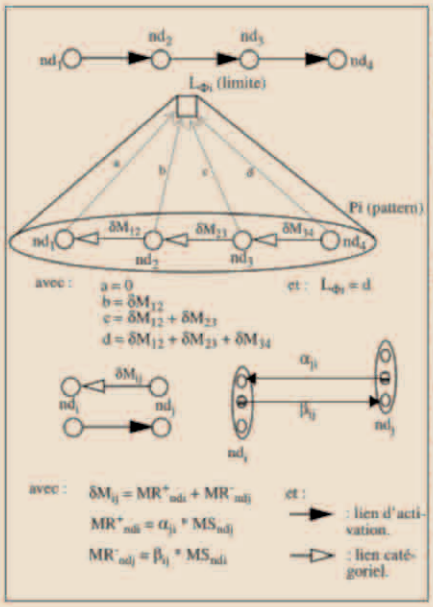


Schéma-4

Une première tentative de description catégorielle de la notion de forme a été faite dans [alb 94] il s'agissait de considérer une forme (Φ) comme une collection de patterns catégoriels dont la limite correspondrait à la variation de quantité de masse relationnelle apportée à la base, consécutivement à l'activation de cette forme ; c'est-à-dire de tous les noeuds qui la composent. Cette limite est la différence entre la somme des quantités de masse relationnelle produites dans cette forme avant son activation et la somme des quantités de masse relationnelle produites par cette même forme après son activation. La limite d'une forme est donc homogène à une quantité de masse relationnelle obtenue par différence de sommations sur des quantités de masse relationnelle. Elle mesure le changement d'état de la forme.

Une nouvelle tentative est en cours[1] ; il s'agit de considérer des formes particulières qui regroupent des entités interchangeables et qui coopèrent entre elles afin de répondre à des sollications externes. Dans l'exemple du partage de ressources (serveurs) développé un peu plus loin, on appèlle catégorie l'ensemble des serveurs qui fournissent un service identique et qui coopèrent entre eux pour satisfaire les besoins d'un certain nombre de clients. La transitivité, propriété nécessaire aux catégories, est traduite ici indirectement par le protocole de recherche des serveurs. Ce protocole transforme chaque requête client/serveur en un agent émissaire qui va de serveur en serveur à travers une ou plusieurs catégories, jusqu'à ce qu'il rencontre le serveur disponible. L'activation du bon serveur en terme de variation de l'activité relationnelle produite, est transitif par rapport aux cheminement des activations infructueuses avec les autres serveurs rencontrés dans la catégorie. Nous verrons d'ailleurs comment et pourquoi, lorqu'un agent émissaire échoue sur un serveur et est renvoyé sur un autre, un lien relationnel est établi entre ces deux serveurs.

Nous aborderons rapidement ce point car il est encore du domaine de la pré-étude, d'autant qu'il pourrait être très utile pour aborder les mécanismes de réorganisation structurelles basés sur la duplication d'objets, étude que nous vennons tout juste de démarrer XX-ref (thèse d'Emmanuel).

Les paramètres qui caractérisent les formes et leurs évolutions relationnelles sont nombreux. On trouve entre autres :

- La variation de masse relationnelle *L(Φ)* due à l'activation de la forme Φ.

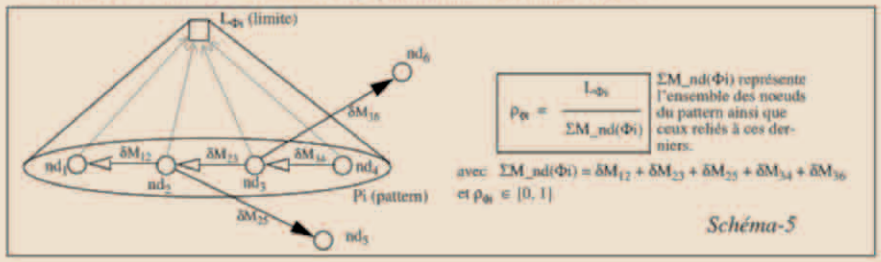


Schéma-5

1. A cette occasion une collaboration est en cours de discussion (stade très préliminaire, par faute de temps) avec un spécialiste des catégories à l'université de Caen.

☛ La *densité relationnelle* ρ_ϕ d'une forme ϕ comme étant le rapport entre L(Φ) et la variation de masse relationnelle issue de l'ensemble des liens et des noeuds touchés par Φ. Comme nous le voyons sur le schéma ci-dessus pour les noeuds nd_2 et nd_3, le pattern P_j représentant la forme Φ_j, n'englobe pas nécessairement tous les liens partant des noeuds de P_j. Ceci traduit le fait que certaines de ces branches ne respectent pas le principe d'homogénéité. La densité relationnelle telle qu'elle vient d'être définie est bien homogène à une constante (pourcentage) et est comprise entre zéro et un.

Cette notion de densité peut être interprétée qualitativement. Appliquée à une forme donnée, elle mesure l'évolution de la complexité interne (structurelle) de cette forme vis-à-vis de son utilisation externe en tant que bloc fonctionnel. Lorsque la densité d'une forme tend vers la valeur "un", quelquesoit l'utilisation externe qui est faite de cette forme, cela signifie que la structure interne de cette forme est complexe en terme de nombre de composant et de liens entre les composants du pattern la représentant. A l'inverse si sa densité tend vers la valeur zéro cela indique que sa structure interne est soit "simple" dans l'absolu, soit "simple" par rapport à l'usage qui en est fait. Qualitativement parlant, la densité d'une forme est donc liée à la complexité de sa structure interne vis-à-vis de son utilisation. D'après cette définition une forme sera dense si elle est très complexe ou si sa complexité, quelle quelle fût, reste grande par rapport à son utilisation. Ce critère qualitatif peut être utilisé pour favoriser l'émergence ou la constitution de certains types de forme.

☛ La *densité relationnelle locale* ($\rho_{ndi,\Phi}$) à un noeud nd_i dans une forme Φ. Il s'agit de mesurer la contribution de chaque noeud rapporté à celle de la forme dont il fait partie.

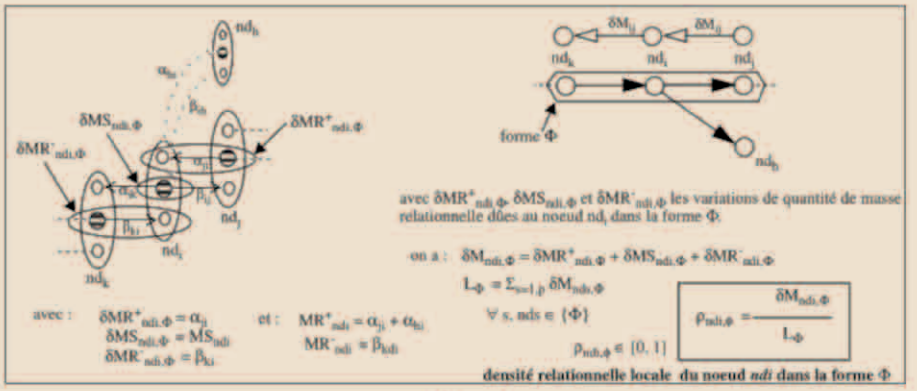
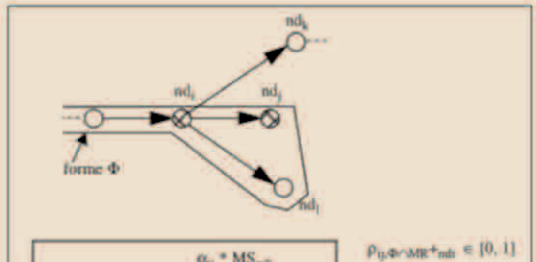


Schéma-6

En étudiant l'évolution de ces contributions locales ont pourra détecter les ruptures d'homogénéité d'une forme. Pour cela il faut reprendre la description interne de chaque noeud à l'intérieur de la forme Φ et sommer pour chacun d'eux dans δM_{ndi} les valeurs d'accroissement de masse relationnelle qui leur sont dûs.

La densité relationnelle locale ($\rho_{ndi,\Phi}$) du noeud nd_i dans la forme Φ est le rapport entre cette quantité δM_{ndi} et la limite (L_Φ) de la forme Φ incluant ce noeud nd_i.



$$\rho_{ij,\Phi\cap MR^{+}ndi} = \frac{\alpha_{ij} * MS_{ndj}}{\Sigma_{p=1..g}(\alpha_{ip} * MS_{ndp})} \qquad \rho_{ij,\Phi\cap MR^{+}ndi} \in [0, 1]$$

avec $\forall$ p, ndp $\in$ {VU (ndi) $\cap$ Φ}

sur la figure il s'agit des noeuds *ndj* et *ndl* ; le noeud *ndk* n'appartient pas à la forme Φ ; g vaut 2.

VU (ndi) est l'ensemble des noeuds reliés à *ndi*, dont *ndi* est le noeud origine.

Schéma-7

☛ La *densité relationnelle d'attraction* ($\rho_{ij,\Phi\cap MR^{+}ndi}$) à un lien nd_i-nd_j dans une forme Φ. Il s'agit de mesurer la contribution du lien entre les noeuds nd_i et nd_j rapportée à celle de la forme dont il fait partie.

Il s'agit d'une densité locale à la fois à un noeud *ndi* et à une forme Φ. Pour cela il faut que le noeud *ndi* appartienne à la forme Φ. Contrairement à la densité relationnelle locale $\rho_{ndi,\Phi}$, il s'agit du rapport entre l'apport en quantité de masse relationnelle dû au lien *ij* et celui dû à la partie de MR^{+}_{ndi} qui est incluse dans la forme Φ. Nous aurions pu prendre MR^{+}_{ndi} entier, c'est-à-dire y compris les liens hors forme ; nous verrons comment cette restriction permet d'augmenter l'impact des formes dans l'évolution des flux entropique et excitatoire.

Comme nous l'avons souligné pour certains, tous ces paramètres ont des interprétations qualitatives particulières. En étudiant l'évolution de ces contributions locales dans le temps, on pourra détecter des ruptures d'homogénéité d'une forme et restucturer le système en conséquence.

Le point de vue global : la base

La base relationnelle (*BCO*) regroupe l'ensemble du graphe dans lequel on observe l'évolution dynamique de formes relationnelles collectives. La base (*BCO*) possède ses propres paramètres relationnels qui caractérisent son état relationnel statique mais aussi dynamique (évolution). Par composition des accroissements de masse relationnel des noeuds ou dans certaines conditions des formes collectives, on obtient la masse de la base et ses variations dans le temps.

Grâce aux paramètres relationnels propres à la base *BCO*, on peut construire de nouveaux paramètres utiles pour les mécanismes évolutionistes intégrés dans chaque noeud. C'est le cas de la densité relationnelle ($\rho_{\Phi,BCO}$) d'une forme Φ dans la base *BCO*. Contrairement à ρ_{Φ} où il s'agissait de définir le rôle du noeud dans la forme correspondante, ici il s'agit de définir le rôle de la forme vis-à-vis de la base.

$$\rho_{\Phi,BCO} = \frac{L_{\Phi i}}{M_{BCO}} \quad \text{avec,} \quad M_{BCO} = \Sigma_{v=1,t}\left[\frac{M_{ndv}}{AD_{ndv}}\right] \qquad \rho_{\Phi,BCO} \in [0, 1]$$

où AD_{ndv} est l'altitude dérivationnelle du noeud nd_v dans le référentiel global d'accessibilité ; c'est la visibilité que BCO a de nd_v ; si AD_{ndv} est très grand alors nd_v est proche de l'oubli définitif pour BCO.

Schéma-8

Cela permet d'étendre le champ des heuristiques qualitatives que l'on pourra introduire dans le modèle relationnel. C'est le cas par exemple pour le calcul des flux entropique et excitatoire que subit chaque noeud pendant son évolution. Un exemple d'heuristique est de faciliter la mémorisation/conservation des liens relationnels dans le cas de bases ne possédant pas beaucoup de connaissances. Réciproquement lorsque de telles bases atteignent un seuil optimal d'absorbtion de connaissances alors les flux excitatoire et entropique en tiennent compte pour freiner la mémorisation/conservation de nouveaux liens relationnels. Ce type d'heuristique est intéressant si la saturation d'une base implique une chute globale des performances du système.

Le recours à ces trois points de vue permet de dissocier les *oublis relatifs* (restructuration) où une information sort du paysage d'une autre information par la rupture du lien entre elles, des *oublis absolus* où une information n'entretient plus aucun lien avec d'autres informations de la base.

Toute entité de la base est décrite dans chacun de ces points de vue à l'aide des paramètres que nous avons présentés. Nous allons voir comment ces paramètres permettent d'intégrer l'évolution des interactions et ce que nous appelons la *topologie relationnelle*, qui va jouer un rôle tout particulier dans la prise en compte d'heuristiques qualitatives au niveau du modèle relationnel lui-même.

On entend par topologie relationnelle d'une entité individuelle ou collective quelconque la structure géométrique de ses interactions avec elle-même ou avec d'autres entités. Pour ne pas entrer dans des explications laborieuses car prématurées, disons simplement que la notion de *forme* permet de passer de l'individu au collectif (ensemble d'entités individuelles en interaction), et que la notion de *topologie* associée à ces formes permet de travailler sur la structure de ces formes. En raisonnant sur ces structures nous prenons en compte dans le modèle relationnel des informations/heuristiques qualitatives.

A titre d'exemple citons l'introduction dans le fonctionnement du modèle relationnel d'heuristiques inspirées de l'*effet d'inférence* ou *fan-effect* XX-ref [cor 87], observé en psychologie cognitive. Cet effet concerne les interférences en situation de reconnaissance. Dans les situations de mémorisation chez l'homme, des tests ont montré que plus il y a d'informations liées à un concept, plus difficile ou lente est la récupération de faits particuliers sur ce concept. Dans l'approche relationnelle la *forme* se compose du concept et de ses faits particuliers ; alors que la structure (topologie) de cette *forme* est définie à partir de caractéristiques comme le nombre de ses faits particuliers.

Pour le placement d'objet dans les systèmes répartis à objets, l'introduction de cette heuristique propre aux mécanismes de mémorisation chez l'homme n'est peut-être pas justifiée ; il est probable que d'autres heuristiques propres au domaine étudié soient mises en évidence. L'important est de montrer comment le modèle numérique proposé pour gérer l'évolution dynamique des interactions, puisse intègrer des heuristiques qualitatives pertinentes aux domaines applicatifs visés.

1.3 . Dynamique des flux excitatoires et entropiques

La description du modèle des interactions dans les trois points de vue "global, collectif et individuel", est une vision objective du système en fonctionnement. Ce modèle est applicable aux systèmes qui fondent leur auto-adaptation sur les phénomènes stables (reproductible dans le temps et dans l'espace) qu'ils produisent. Cela signifie que si le système observé ne possède aucune régularité comportementale intrinsèque (système chaotique) dans la période d'observation fixée, au mieux le modèle relationnel pourra dire qu'il ne voit pas de régularité. En revanche, le système peut présenter des régularités intrinsèques mais non explicitées ; dans ce cas le modèle peut aider le système à modifier sa propre structure opérationnelle pour révéler ces régularités.

Je prendrai comme exemple un ensemble d'utilisateurs et un "pool" d'imprimantes à leur disposition. La répartition géographique des imprimantes (structure opérationnelle) peut laisser à penser, de prime abord, que le comportement des utilisateurs (contexte) n'est pas stable dans le choix d'une imprimante (on exclue le comportement stable qui consiste à choisir une imprimante au hasard !). En réalité, il suffit de changer la disposition des imprimantes dans le batîment (structure du système observé) pour que le comportement des utilisateurs révèle la stabilité intrinsèque du système. Elle peut s'énoncer par le fait que chaque utilisateur a besoin d'une imprimante principale et d'une éventuelle seconde en secours.

En introduisant des heuristiques qualitatives à la fois dans le modèle relationnel pour faciliter la vision des formes stables, mais aussi dans les stratégies de réorganisation opérationnelle pour modifier le système, on peut faire émerger certaines stabilités potentielles du système que sa configuration initiale ne permettait pas d'exister.

Rôles des flux entropique et excitatoire.

L'intensité des liens qui relient les noeuds entre eux dépend principalement des activations réellement faite entre ces noeuds. C'est le domaine d'application choisi qui précise la notion d'activation ; dans le cas des clients/serveurs en informatique, l'activation correspond à l'émission d'une requête de service entre un client et un serveur. Plus un noeud active un autre noeud et plus le coefficient (intensité relationnelle XX-ref [bou 92]) attaché au lien correspondant est grand.

Le flux excitatoire précise la façon dont croît ce coefficient. Pour cela nous allons préciser comment il intègre les heuristiques qualitatives qui tiennent compte des paramètres relationnels du contexte suivant les différents points de vue locaux, collectifs et éventuellement globaux définis précédemment. Parmi ces paramètres on trouve ceux (comme $\rho_{ij,\Phi r \cap MR^+ndi}$) qui sont liés à la composition relationnelle du noeud *ndi* , ceux (comme $\rho_{ndi,\Phi}$) qui sont liés à la structure topologique de la forme collective dans laquelle évolue potentiellement ce noeud et éventuellement ceux (comme $\rho_{\Phi,BCO}$) qui sont liés à la base (*BCO*) elle-même.

Pour simplifier cette étude nous ne ferons allusion au référentiel global d'accessibilité et aux mécanismes associés, que dans le cadre de la description des mécanismes liés aux référentiels locaux d'accessibilité. Sans revenir en détails sur XX-ref [bou 92], on peut rappeler simplement que la notion de référentiel local d'accessibilité traite de la vision individuel de chacun des noeuds de la base en interaction avec les autres. Lorsqu'un noeud interagit dans la base il mesure ses interactions dans le référentiel local d'accessibilité qui lui est propre. A l'opposé, le référentiel global d'accessibilité représente le point de vue global, c'est-à-dire celui de la base. L'introduction des formes collectives permet de suivre la dynamique des interactions entre les noeuds, c'est en quelque sorte un niveau intermédiaire fortement dynamique entre le point de vue individuel (noeud) et le point de vue global (base) ; ou comment les individualités participent à une certaine globalité via des actions collectives limitées.

L'usage du point de vue local et des formes collectives intermédiaires permet de gérer les *oublis* relatifs des individus entre eux. L'oubli signifie pour un noeud donné, la disparition de son référentiel local d'accessibilité d'un autre noeud ; cela ne préjuge en rien sur les autres liens que peut entretenir ce noeud oublié avec d'autres noeuds de la base. En fait à ce niveau plutôt que de parler d'oubli, on parlera de restructuration.

Alors que l'usage du point de vue global permet entre autre de gérer l'oubli collectif, celui au niveau de la base. Un noeud sera oublié pour la base lorsqu'il n'entretiendra plus aucun lien avec aucun autre de ces condisciples. C'est cet aspect de l'oubli réel qui ne sera pas abordé dans ce texte ; par contre on envisagera ici le point de vue global pour parler du rôle que peut jouer la masse de la base dans certains phénomènes comme par exemple celui de l'évolution dynamique des flux excitatoire et entropique.

Le graphe relationnel de la base est composé de noeuds et de liens relationnels entre ces derniers. Deux noeuds sont reliés par un lien dans ce graphe s'ils ont entretenu dans le passé et/ou entretiennent des relations d'activation entre eux. Nous verrons plus loin comment l'entropie joue un rôle notamment dans la perrenité de ces liens dans le temps, en particulier si ces noeuds n'entretiennent plus de relation d'activation entre eux. Un lien d'activation entre deux noeuds correspond à l'activation par l'un des noeuds de l'autre noeud. On parlera indifféremment d'activation, de stimulation ou encore d'excitation dans une pareille situation.

On verra également que ces liens définissent la mémoire de l'évolution du comportement du système dans son processus d'auto-adaptation. Dans l'exemple de l'optimisation de la production, proposé en IV-III, on verra plus précisément comment un lien peut survivre aux entités (noeuds) qui l'ont créé et comment ces mêmes liens jouent un rôle actif dans des processus explicatifs ou liés au recouvrement d'une situation antérieure (par exemple en cas d'agrégation dans les mécanismes de duplication).

Contrairement au phénomène entropique, l'excitation est un phénomène potentiellement propagatoire ; en effet l'activation d'un noeud par un autre noeud, peut conduire ce dernier à lui-même en activer un ou plusieurs autres. Dans [bou 92] nous avons distingué les excitations directes qui agissent directement sur tout noeud activé, des excitations indirectes qui sont la répercussion des excitations directes au niveau des référentiels locaux d'accessibilité. De la même manière l'entropie se décompose en une entropie locale à chaque référentiel local d'accessibilité et en une entropie globale à la base. La suite de cette étude se focalisera donc sur l'excitation indirecte (*EXCI*) et sur l'entropie locale (*ENTL*). Nous n'allons pas reprendre tout ce qui a été présenté dans [bou 92] à propos des flux entropique et excitatoire, mais profiter des deux paragraphes suivants pour montrer comment le modèle relationnel peut prendre en compte certaines heuristiques qualitatives (masse d'interaction et topologie des formes relationnelles collectives) dans les mécanismes évolutionnistes qu'il propose. Nous regarderons donc plus particulièrement les mécanismes locaux et les flux associés.

Nous présentons dans cette étude deux approches différentes pour aborder la réalisation de l'intégration des heuristiques qualitatives dans un tel modèle numérique.

L'une d'entre elle est partiellement centralisée, alors que l'autre est entièrement répartie. L'approche *partiellement centralisée* manipule des informations (paramètres) sur des entités collectives ou globales comme la masse relationnelle de la base, celle (limite) d'une forme relationnelle collective ou encore les densités qui en découle. Nous verrons dans cette approche comment il est possible de prendre en compte dans le modèle d'attraction, les quantités de masses relationnelles contextuelles. Par contre cette approche nécessite de connaître des informations globales et de les utiliser dans des calculs locaux. Outre les problèmes théoriques de propagation de ces informations en terme de temps de calcul ..., cette approche nécessite le recours à des entités externes aux entités observées qui devront mettre à jour (par diffusion ou autre mécanisme) ces informations globales auprès des entités locales (dans l'exemple présenté en IV-III nous avons eu recours à un objet "Manager" qui centralise par exemple la charge globale du domaine qu'il gère).

L'autre approche strictement *répartie* consiste à limiter l'impact des entités collectives à la vision que peut en avoir chaque entité individuelle. Les formes existent de la même façon quelquesoit l'approche considérée, néanmoins les calculs travailleront sur des parties locales à ces formes. Plutôt que de manipuler la densité d'un noeud dans l'ensemble de la forme qui l'englobe, on prendra des informations sur ce noeud dans le contexte qui le touche directement. Nous verrons comment cette approche privilégie davantage la topologie du contexte local propre à chaque individualité. Contrairement à l'autre approche, il sera difficile d'intégrer dans cette approche répartie, l'impact direct de la masse relationnelle du contexte des entités dans le modèle d'attraction.

Les deux approches devraient être étudiées expérimentalement et par simulation, afin d'en mesurer la faisabilité, le coût et les performances. après avoir présenté les deux approches, nous limiterons les développements de ce texte à l'approche répartie, c'est-à-dire celle qui n'intègre pas les propriétés des formes collectives et celles de la base.

i) Excitation - Stimulation - Activation

L'excitation indirecte ou locale consiste à faire varier le coefficient du lien reliant le noeud excitateur au noeud excité. Cette variation tient compte d'un certain nombre de paramètres qui sont traduit dans ce que l'on appelle le flux excitatoire local. Parmi ces paramètres on trouve la composition relationnelle du noeud excitant, la structure topologique du référentiel local d'accessibilité correspondant à ce noeud excitant, celle de la forme éventuelle dans laquelle évolue ce noeud excitant et enfin des paramètres liés à la base. L'évolution de ce coefficient renseigne sur le pouvoir d'attraction relationnel que possède le noeud origine du lien sur les noeuds qu'il active. Suivant le domaine applicatif visé, on peut considérer différents modèles d'attraction. Dans le domaine du placement d'objets informatiques entre des clients et des serveurs, le modèle d'attraction choisi est celui des serveurs vers les clients. Lorsqu'un client utilise un serveur, c'est le serveur qui est attiré par le client et non l'inverse (nous verrons dans 4-III comment des heuristiques plus fines sont utilisées).

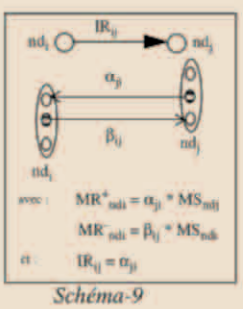


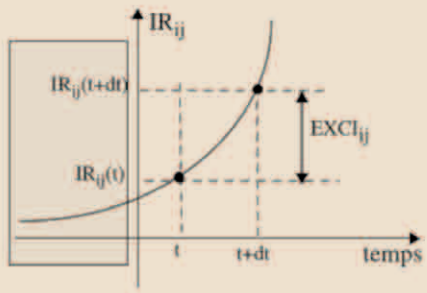
Schéma-9

Un lien relationnel entre deux noeud nd_i et nd_j, est définit par deux coefficients (α_{ji} et β_{ij}). Le premier coefficient α_{ji} représente justement l'attraction de nd_j vers nd_i ; il est valide dans le référentiel local d'accessibilité du noeud nd_i qui active le lien. Le second coefficient β_{ij} contribue lui à la variation de quantité de masse relationnelle du noeud nd_j consécutivement à cette activation du lien entre nd_i et nd_j. Ce coefficient est valide dans le référentiel local d'accessibilité de nd_j et par conséquent ne joue pas de rôle dans le calcul du flux excitatoire lié à ce lien (nd_i - nd_j). C'est la raison pour laquelle nous ne définissons pas de flux excitatoire sur le lien β_{ij}. Nous posons α_{ji} comme étant égal à IR_{ij} défini dans [bou-a 92] ; cette grandeur est appelée l'intensité relationnelle depuis nd_i vers nd_j.

L'intensité relationnelle *(IR_{ij})* est calculée continuement par rapport au temps, contrairement à la masse relationnelle qui est calculée discrètement à la fin de chaque période d'observation. L'intensité relationnelle est calculée au temps t en fonction de sa valeur au temps

t-δt. La variation obtenue sur δt est appelée la fonction d'excitation indirecte *EXCI* appliquée à ce lien du référentiel local. Au sein d'un référentiel local donné, chaque lien possède sa propre fonction *EXCI* ; l'étude de chacune de ces fonctions *EXCI* permet de comparer l'évolution des liens dans ce référentiel local ; on parle par exemple de compression ou de dilatation locale.

Comme nous l'avons dit, l'intensité relationnelle (IR_{ij}) caractérise l'intensité du lien issu des interactions entre deux noeuds (nd_i et nd_j). Cette grandeur est bornée par une valeur minimale, *IR_min* (0 sur le schéma ci-contre) et par une valeur maximale, *IR_MAX*. Lorsque deux noeuds interagissent pour la première fois *IR* vaudra *IR_min* ; une très forte interaction entre deux noeuds conduira à faire tendre *IR* vers *IR_MAX*. Tout l'intérêt des flux (excitatoire et entropique) est justement dans la manière dont *IR* va évoluer entre ces deux valeurs extrêmes.



$$IR_{ij}(t) = inf((IR_{ij}(t) + EXCI_{ij} \times dt), IRMAX)$$

Dans (bou-a 92) nous avions proposé que l'excitation (*EXCI*) dépende (g_3) de l'intensité relationnelle (*IR*).

$$EXCI_{ij} = g_3(IR_{ij})$$

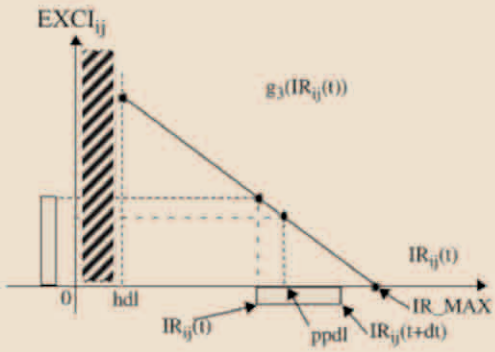


Schéma-10

Cela nous permettait d'introduire des comportements différents au niveau du système relationnel. Par exemple on souhaitait que les interactions aient de plus en plus d'effet sur *IR* au fur et à mesure que *IR* se rapprochait de *IR_MAX*. Une interprétation qualitative (heuristique comportementale) de ceci peut être que le système a tendance à aggréger les composants qui interagissent ; on parlera d'un système conservateur. Nous verrons plus en détails un peu plus loin, les grands principes qui régissent tous les comportements possibles qui sont offerts par la combinaison des flux excitatoire et entropiques.

En choisissant une fonction affine (a_1, b_1) pour g_3, *IR* devient une fonction exponentielle du temps. Avec :

$$g3(IRij) = a_1 IR_{ij}(t) + b_1$$

on obtient :

$$IR_{ij}(t) = k \times e^{a_1 t} - \frac{b_1}{a_1}$$

où k est une constante.

Dans (bou-a 92) nous avions appelé "variation réelle de l'excitation (indirecte)" *EXCI'*, pour le lien entre les noeuds nd_i et nd_j au temps t, la pente de g_3. Nous parlions alors pour cette grandeur de "flux excitatoire (indirect)" lié au lien relationnel ndi-ndj, caractérisé par IR_{ij}.

La construction de cette fonction (*EXCI'*) est très importante pour l'introduction d'heuristiques qualitatives dans le modèle relationnel, puisqu'elle caractérise en quoi dépend l'évolution de la variation de *IR*. Nous avons proposé dans (bou-a 92) de relier cette grandeur à une fonction (g_4) de la masse relationnelle de nd_i (noeud origine de l'interaction). Cela permettait au référentiel local de jouer un rôle dans l'évolution de cette grandeur. En particulier on pouvait ainsi introduire dans le modèle la notion d'attraction relationnelle ; plus un noeud joue un rôle relationnel important (nombreuses interactions avec beaucoup d'autres noeuds) plus il a une masse relationnelle importante et plus l'intensification de ses liens d'interactions est "efficace".

$$EXCI'_{ij} = \frac{d}{dIR}(EXCI_{ij}) = \frac{d}{dIR}g_3(IR_{ij}) = a_1$$

$$EXCI'_{ij} = g_4(M_i)$$

Les récents travaux que nous avons faits (bou-a 97) pour explorer les paramètres caractéristiques des trois points de vue (global, collectif et individuel) nous ont amené à manipuler la notion de densité. Par exemple la densité relationnelle d'attraction (ρ_{ij+}) au lien nd_i-nd_j renseigne sur l'importance de ce lien pour le noeud nd_i par rapport aux autres liens qu'il entretient avec d'autres noeuds. Dans l'exemple de l'ajustement de la production des serveurs, ce paramètre renseigne sur l'importance pour un client donné de tel ou tel serveur avec lesquels il fonctionne. Ceci nous a amené à définir g_4 non plus directement à partir des masses relationnelles, comme dans (bou-a 92), mais plus finement à partir de ces nouveaux paramètres qui utilisent toujours les masses relationnelles, mais à travers les densités. Cela normalise le résultat de g_4 et facilite comme nous le verrons, l'interprétation des équations résultantes.

A cette notion d'attraction nous avions pensé introduire dans le calcul de g_4, la notion de topologie relationnelle. Il s'agit en effet de tenir compte dans son calcul de la forme topologique du contexte relationnel dans lequel sont effectuées les interactions. Un exemple de description topologique d'un objet relationnel est donné par le nombre de liens du référenciel local d'accessibilité définit par cet objet. Mais on peut décrire la topologie de la forme relationnelle dans laquelle s'inscrit un objet en allant au-delà du point de vue individuel (référentiel local) ; en particulier on peut regarder la topologie de la forme collective stable qui inclut cet objet ; on peut même regarder celle de la base qui englobe cette forme collective. Cela permet de pondérer l'attraction relationnelle par des coefficients qui renseignent sur ces différentes topologies. Comme nous le verrons dans les formules proposées, on peut pondérer cette pondération pour modifier l'influence de la topologie en fonction du point de vue considéré. Dans les exemples de formules proposés on élève au carré le terme (*B*) qui provient de la forme collective et au cube celui (*C*) qui provient de la base. Ces termes étant compris dans l'intervalle (-1, 1), on minimise ainsi l'impact de la forme collective vis-à-vis du référentiel local et celui de la base vis-à-vis de la forme collective.

On retrouve ici une façon d'intégrer au comportement du système, des heuristiques issues de travaux de psychologie cognitive (cor 87), et qui concernent essentiellement l'*effet d'interférence* et le *fan-effect* (effet d'interférence en situation de reconnaissance). Ces travaux supposent que plus il y a d'informations (nombre) liées à un concept, plus difficile ou lente est la récupération en mémoire de faits particuliers sur ce concept. Comme nous le soulignions dans (bou-a 92), au-delà de la pertinence de ces notions transposées dans des domaines très différents de ceux qui les ont produits, il paraît intéressant de voir que l'on peut manipuler ce type d'heuristiques au niveau du modèle relationnel. Concrètement nous allons voir une illustration de ces heuristiques dont la transposée apporte des choses intéressantes dans l'exemple présenté de l'allocation des ressources dans un environnement de type client-serveur. Nous verrons par exemple qu'une transposition pertinente de ces comcepts est de considérer que plus un client utilise de façon équi-probable un nombre important de serveurs, moins il mémorisera l'un d'entre eux. Une des conséquences opérationnelles de ceci est qu'à chaque fois que le client à besoin de l'un des serveurs en question, il lancera un processus de choix ouvert sur n'importe lequel d'entre eux. Ce processus de choix est multiple ; cela peut être un tirage aléatoire (cf. le mode de diffusion dans les groupes COOL au chapitre 3-III) sur l'un des serveurs connus ou encore la création d'un agent (cf. 4-III) requête autonome, disposant de moyens de recherche et de négociation. A l'inverse si le nombre de ces serveurs est petit, voir tend vers 1, alors le client aura intérêt à mémoriser ce serveur pour interagir directement avec lui (dans ce cas la récupération en mémoire est très bonne).

Une approche partiellement centralisée

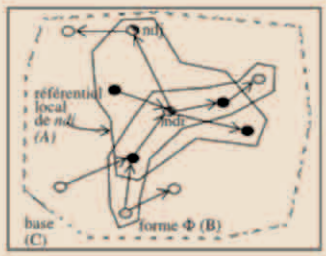
L'approche partiellement centralisée consiste tout simplement à faire intervenir les trois points de vue individuel, collectif et global, dans le calcul des flux excitatoire (*EXCI'*) et entropique (*ENTL'*). Dans ce cas il faut que chaque noeud ait accès à des informations sur les formes collectives (masse relationnelle, topologie relationnelle ...) et sur la base. En excluant les termes liés aux formes collectives et à la base, dans le calcul des flux, l'autre approche (répartie) ne nécessite pas l'accès à des informations centralisées.

Le flux excitatoire indirect EXCI' se compose donc des trois termes *A* (noeud), *B* (forme) et *C* (base) ; chacun d'eux prend en compte les trois niveaux d'influence (abstractions) que l'on cherche à intégrer dans ce calcul. Nous allons voir terme par terme comment chacun d'eux contribue à l'augmentation des écarts relationnels existant entre les liens d'un même référentiel local, tout en favorisant la stabilité de certains liens. On retrouve la même chose dans le cas du flux entropique. Nous montrerons comment la combinaison de ces flux contribue à des comportements globaux du système, qui sont interprétables qualitativement.

Nous distinguons le cas (a) où le noeud nd_i (origine de l'interaction) est inclus dans une forme collective Φ, du cas (b) où nd_i n'appartient à aucune forme Φ de la base. Nous verrons comment les formules proposées pour les deux cas, sont identiques lorsque la forme collective qui inclut le noeud visé est la base toute entière (Φ -> *BCO*).

Dans la mesure où g_4 (*EXCI'*) dépend de densités de masses relationnelles comprises dans l'intervalle (0, 1), et pour normaliser le paramétrage du système via les flux, nous avons introduit dans g_4 un coefficient (1/3 pour le cas "a" et 1/2 pour le cas "b") qui ramène les valeurs possibles de g_4 dans l'intervalle (-1, 1). Pour cette même raison, les fonctions utilisées pour intégrer au calcul des flux les caractéristiques des différentes topologies relationnelles des entités concernées, sont définies de N^{+*} dans (-1, 1).

a) Le noeud ndi est inclus dans une forme Φ



Cas (a) : le noeud *ndi* appartient à une forme Φ.

où :

EXCI'ij est le flux excitatoire du lien de *ndi* vers *ndj*
Φ la forme relationnelle collective dont *ndi* fait partie
BCO la base étudiée

$$EXCI'_{ij} = g4 = \frac{1}{3}\left[A + B^2 + C^3\right]$$

$$EXCI'_{ij} = \frac{1}{3} \times \left[\left(\rho_{ij,\Phi \cap MR^+ ndi} \times h_1\left(\frac{1}{n}\right)\right) + \left(\rho_{ndi,\Phi} \times k_1\left(\frac{1}{m}\right)\right)^2 + \left(\rho_{\Phi,BCO} \times w_1\left(\frac{1}{p}\right)\right)^3\right]$$

avec :
n = nombre de liens dont nd_i est origine et dont l'extrémité est un noeud de Φ.
m = nombre de noeuds de Φ
p = nombre de formes de la base
$h_1()$, $k_1()$ et $w_1()$ sont définies de N^{+*} dans [-1, 1]

$$\rho_{\Phi,BCO} = \frac{L_\Phi}{M_{BCO}} \quad \text{avec,} \quad M_{BCO} = \Sigma_{v=1,z}\left[\frac{M_{ndv}}{AD_{ndv}}\right]$$

Rappel

$$\rho_{ndi,\Phi} = \frac{\delta M_{ndi,\Phi}}{L_\Phi}$$

$$\rho_{ij,\Phi \cap MR^+ ndi} = \frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}(\alpha_{ip} * MS_{ndp})}$$

avec ∀ p, ndp ∈ {VU (ndi) ∩ Φ}

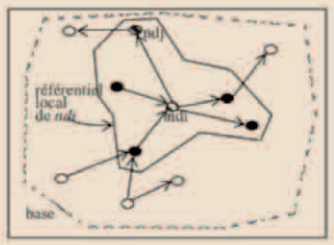
La formule proposée pour *EXCI'* (g_4) est un exemple possible parmi d'autres. Celle-ci conduit aux interprétations suivantes.

Dans le cas du terme *A* concernant le référentiel local d'accessibilité du noeud nd_i, on modifie linéairement l'intensité du flux de façon inversement proportionnelle au nombre de liens dont nd_i est origine et dont les extrémités sont des noeuds de la forme Φ. Dans ce cas, cela signifie que le phénomène d'attraction liée à l'importance de la masse de nd_i, est d'autant plus tassé que nd_i active beaucoup d'autres noeuds dans cette forme.

Pour le terme *B* concernant l'influence de l'éventuelle forme à laquelle appartient nd_i, ce tassement est inversement proportionnel au nombre de noeuds de cette forme. Cela peut être interprété comme étant une diminution de l'influence d'un noeud dès lors qu'il appartient à une forme complexe en nombre de liens.

Enfin pour le terme *C* concernant l'influence de la topologie de la base sur le flux excitatoire local au lien courant, le tassement est inversement proportionnel au nombre de formes dans la base. On peut également prendre le nombre de noeud dans la base ; des études expérimentales ou en simulation devront être menées pour préciser les influences respectives de ces choix.

b) Le noeud nd_i n'est pas inclus dans une forme Φ



Cas (b) : le noeud *ndi* n'appartient pas à une forme Φ.

où :

EXCT'ij est le flux excitatoire du lien de *ndi* vers *ndj*
Φ la forme relationnelle collective dont *ndi* fait partie
BCO la base étudiée

$$EXCI'_{ij} = g4 = \frac{1}{2}\left[A + B^2\right]$$

$$EXCI'_{ij} = \frac{1}{2} \times \left[\left(\frac{\alpha_{ji} \times MS_{ndj}}{MR^{+}_{ndi}}\right) \times h_1\left(\frac{1}{n}\right) + \left(\frac{M_{ndi}}{M_{BCO}} \times k_1\left(\frac{1}{m}\right)\right)^2\right]$$

Schéma-11

Contrairement au cas précédent (a), ici il n'existe pas de terme provenant de l'influence d'une forme incluant ce noeud. De même les densités utilisées couvrent l'ensemble du référentiel local de nd_i.

Terme A : influence du référentiel local

On voit sur le schéma ci-contre que les termes A_a et A_b respectivement issus de la formule de calcul de *EXCI'* pour les cas (a) et (b), sont identiques lorsque la forme stable qui englobe ndi est la base entière.

$$A_a = \left[\rho_{ij,\Phi \cap MR^+_{ndi}} * h\,(1/n) \right] \quad \text{cas (a)} \qquad A_b = \left[\left(\frac{\alpha_{ji} * MS_{ndj}}{MR^+_{ndi}} \right) * h\,(1/n) \right] \quad \text{cas (b)}$$

$$\rho_{ij,\Phi \cap MR^+_{ndi}} = \frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})}$$

avec $\forall$ p, ndp $\in$ {VU (ndi) $\cap$ Φ} VU (ndi) est l'ensemble des noeuds reliés à *ndi* et dont *ndi* est le noeud origine.

$$\lim_{\Phi \to BCO} (VU(ndi) \bigcap \Phi) = VU(ndi) \quad \Rightarrow \quad \lim_{\Phi \to BCO} \left(\sum_{p=1}^{n} \alpha_{ip} \times MS_{ndp} \right) = MR^{+}_{ndi}$$ on a donc,

$$\lim_{\Phi \to BCO} (\rho_{ij,\Phi \cap MR^+_{ndi}}) = \frac{\alpha_{ji} * MS_{ndj}}{MR^+_{ndi}} \quad \Rightarrow \quad \lim_{\Phi \to BCO} A_a = A_b$$

Il s'agit de regarder l'influence de la topologie du référentiel local d'un noeud donné, dans l'évolution des flux excitatoires qu'engendre ce référentiel local d'accessibilité. Nous avons vu que le terme *A* comme les termes *B* et *C* ont des valeurs comprises dans l'intervalle (-1, 1). L'un des objectifs recherché est de stabiliser les formes ; pour cela plusieurs techniques sont envisageables à partir de ces formules. Suivant l'heuristique comportementale recherchée, on combinera telle ou telle technique. L'une d'entre elles consiste à augmenter les écarts en préservant les liens forts et à limiter le resserrement des liens d'autant plus que les topologies locales sont complexes.

Nous allons classer en deux catégories les principaux cas rencontrés et voir comment pour chacun d'eux ces formules sont conformes aux heuristiques choisies. Le premier cas est celui ou les noeuds activables par le noeud courant sont activés de façon équivalente dans la période d'observation ; dans ce cas les valeurs des flux excitatoires sont sensiblement voisines pour chacune des branches du référentiel, dans une période donnée. L'autre cas est celui où il n'y a pas cette équirépartition des activations entraînant des valeurs voisines entre les flux. Ce deuxième cas peut alors être lui-même subdivisé en deux sous-catégories :

- ☛ celle où la branche courante étudiée est beaucoup plus active que la somme des activités des autres branches activées par le noeud origine ;
- ☛ celle où l'activité de la branche courante est nettement moins active que celle de cer-

taines autres branches activées par le noeud origine.

$$A = \left[\rho_{ij,\Phi \cap MR^{+}_{ndi}} * h\,(1/n) \right] = \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} \right] * h\,(1/n)$$

Avec n, étant égal au nombre de noeuds extrémités des liens partant du noeud courant.

① *Equirépartition des activations depuis nd_i*

$\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp}) = n * (\alpha_{ji} * MS_{ndj}) \Rightarrow A = (1/n) * h\,(1/n) = h\,(1/n^2)$

$\lim_{n \to \infty} A = 0$ si $n = 1$ alors $A = 1$

Commentaire : Plus *n* est grand, moins la variation du flux excitatoire est grande, ce qui est conforme au principe de complexité. Nous reviendrons sur ces interprétations en prenant l'exemple des clients et des serveurs, après avoir parlé de l'entropie.

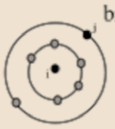
② *Non équirépartition des activations depuis nd_i*

❶ la branche courante (ij) est nettement plus active que l'ensemble des autres branches :

$$\lim \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} \right] = 1 \Rightarrow \lim A = 1$$

Commentaire : Dans ce cas on a une très forte variation du flux ce qui renforce nettement ce lien.

❷ la branche courante (ij) est nettement moins active que certaines autres branches :



$$\lim \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} \right] = 0 \Rightarrow \lim A = 0$$

Commentaire : Dans ce cas on a une très faible variation du flux ce qui détend ce lien relativement aux autres.

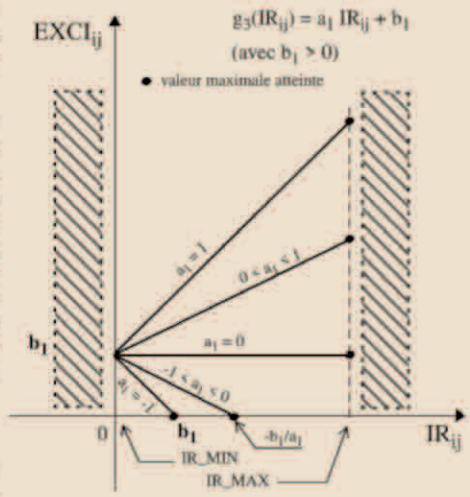
Schéma-12

Revenons sur la fonction d'excitation *EXCI* qui augmente la valeur du lien relationnel nd_i-nd_j à chaque interaction entre les noeuds nd_i et nd_j. Puisque cette fonction (g_3) est une fonction affine (a_1, b_1) de IR, et en faisant abstraction des termes *B* et *C* définis précédemment, sa pente ($a_1 = EXCI'$) doit être comprise entre -1 et 1.

on a l'un des deux cas suivants : $(EXCI_{ij} \geq 0$ *et* $a_1 \in [0,1])$ *ou* $(EXCI_{ij} \geq 0$ *et* $a_1 \in [-1,0])$

Pour simplifier l'étude nous posons que b_1 est supérieur à zéro ($b_1 > 0$). Symétriquement nous verrons que b_2 pour l'entropie est pris inférieur à zéro ($b_2 < 0$) et qu'en plus nous choisirons ($b_1 = - b_2$).

Contrairement au flux entropique qui agit sur les liens, comme nous allons le voir, à chaque intervalle (dt) de temps, le flux excitatoire agit sur l'intensité relationnel d'un lien uniquement au moment où ce lien est activé. Cette précision faite, nous allons regarder les différents cas possibles ($a_1 < 0$, $a_1 = 0$ et $a_1 > 0$) concernant le flux excitatoire. Il ne s'agit plus, comme nous venons de le faire avec le calcul de a_1, de faire une interprétation des conditions qui ont conduit à ce que a_1 ait telle ou telle valeur ; il s'agit de regarder les propriétés du système en fonction des différentes valeurs possibles de a_1.



1 *EXCI* est négatif ($-1 < a_1 < 0$)

Par définition g_3 est positive ou nulle ; en effet toute interaction augmente la valeur du lien correspondant. Choisir ($a_1 < 0$) impose ($b_1 > - a_1 * IR$). On peut prendre, comme le montre le schéma ci-dessus ($b_1 < IR\_MAX$) ; dans ce cas la limite de IR est b_1 pour ($a_1 = -1$) et ($- b_1/a_1$) pour ($-1 < a_1 < 0$). Dans ce dernier cas, si ($a_1 = -1/n$), alors (cf. l'encadré ci-dessous) la limite de IR, lorsque le nombre d'excitations tend vers l'infini, est ($n*b_1$).

Soit E_p, la p^{ième} excitation/interaction sur le lien nd_C-nd_P. Si l'on suppose que l'entropie (*ENTL*) n'a pas d'effet, on peut considérer que IR_{ij} est une fonction non plus du temps t mais des interactions. Nous écrivons donc $IR_{ij}(E_p)$ comme étant la valeur de IR_{ij} consécutivement à l'excitation E_p.

$$IR_{ij}(E_{p+1}) = IR_{ij}(E_p) + EXCI_{ij}$$

$$EXCI_{ij} = a_1 IR_{ij}(E_p) + b_1 \quad avec \quad EXCI_{ij} \geq 0$$

ceci $\Rightarrow$ $\boxed{IR_{ij}(E_{p+1}) = (a_1 + 1) IR_{ij}(E_p) + b_1}$ (i)

si $\boxed{a_1 = -1 \quad \Rightarrow \quad IR_{ij}(E_{p+1}) = b_1 \quad \forall p}$ (ii)

si $-1 < a_1 < 0$

posons $IR_{ij}(E_{p+1}) = y_l$ avec $k = a_1 + 1$

on peut réécrire (i) :

$$y_1 = k y_0 + b_1$$

$$y_2 = k^2 y_0 + k b_1 + b_1$$

$$y_l = k^l y_0 + (k^{l-1} + k^{l-2} + \ldots + 1) b_1$$

$$\boxed{y_l = k^l y_0 + \left(\frac{k^l - 1}{k - 1}\right) b_1} \quad (iii)$$

puisque $-1 < a_1 < 0$,posons $a_1 = -\frac{1}{n}$ avec $n \in N^{+*}$

(suite →)

avec $k = 1 + a_1 = \frac{n-1}{n}$

Nous obtenons à partir de (iii) :

$$IR_{ij}(E_{p+l}) = \left(\frac{n-1}{n}\right)^l \times IR_{ij}(E_p) - \left(\left[\left(\frac{n-1}{n}\right)^l - 1\right] n \times b_1\right)$$

puis

$$\boxed{IR_{ij}(E_{p+l}) = \left(\frac{n-1}{n}\right)^l (IR_{ij}(E_p) - nb_1) + nb_1} \quad (iv)$$

alors de (iv), on obtient :

$$\boxed{\lim_{l \to \infty} IR_{ij}(E_{p+1}) = nb_1} \quad (v)$$

avec l étant le nombre d'interactions.

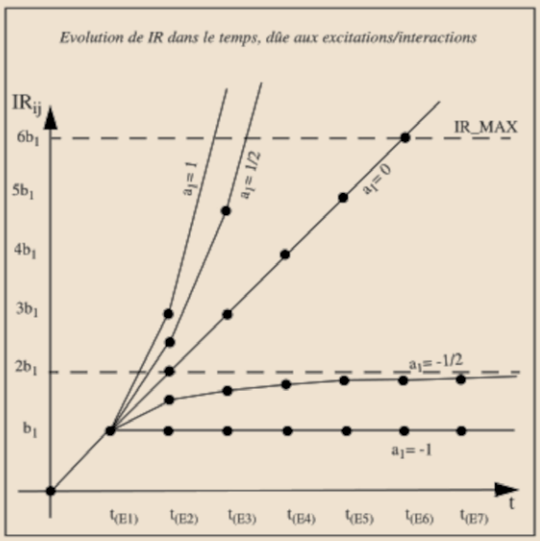
en particulier si $a_1 = -1/2$ *(n=2) on a*

$$\lim_{l \to \infty} IR_{ij}(E_{p+1}) = 2b_1$$

On peut interpréter ce cas en disant que plus *IR* est intense (proche de sa limite) et moins on acroît cette intensité ; inversement, moins le lien est intense (faible interaction) plus ce lien s'intensifie lors d'une interaction. Cela traduit le fait que les valeurs des liens relationnels issus des interactions, convergeront de moins en moins vite vers leur valeur limite.

Cela traduit le fait que l'accroissement de *IR* est borné par ($- b_1/a_1$), et la valeur de IR par ($n*b_1$). Plus a_1 est petit et plus la limite maximale sera faible (pour $a_1 = -1$ on a $IR = b_1$ en une interaction). Dans ce cas le système est très réactif aux interactions. Si l'on couple ce comportement avec une entropie elle aussi très réactive, on pourra parler de la grande élasticité résultante du système.

Evolution de IR dans le temps, dûe aux excitations/interactions



L'interprétation de ce cas est que plus a_1 tend vers -1 et plus on va vers un système où l'effet de l'excitation est binaire, dans la mesure où une seule interaction provoque le seul changement d'état possible de *IR* en passant de *IR_MIN* (0 sur la figure) à b_1.

2) *EXCI'* est nul ($a_1 = 0$)

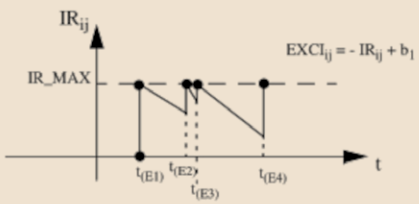
Dans ce cas, à chaque interaction, *IR* augmente de b_1. Si cette constante (b_1) vaut 0, alors quelque soit la valeur de *IR*, le flux excitatoire est sans effet sur le système.

3 *EXCI'* est positif ($0 < a_1 < 1$ et $a_1 = 1$)

L'interprétation de ce cas est que plus *IR* est intense (proche de *IRMAX*) et plus on accroît cette intensité ; et inversement, moins le lien est intense (faible interaction) moins ce lien s'intensifie lors d'une interaction. On peut parler d'un système conservateur et froid, car il garde les liens forts et ne favorise pas l'émergence de nouveaux liens. On parlera

également dans ce cas de faible élasticité du système. La formule de *IR* est équivalente à (*iv*), avec *(n+1)/n* à la place de *(n-1)/n*. Dans ce cas lorsque le nombre d'interaction tend vers l'infini, *IR* tend vers l'infini.

En résumé on voit que plus a_1 est grande (tend vers 1) et plus la fonction *EXCI* fait converger rapidement (en nombre d'interactions) *IR* vers sa valeur maximale (*IR_MAX*). On dira que le système mémorise "vite" et "bien" (on minimise le nombre d'interaction pour une intensité du lien de plus en plus grande). Inversement quand a_1 tend vers sa valeur minimale (-1) la convergence est de moins en moins rapide. On dira que le système mémorise "difficilement" ; on parlera même d'un système réfléchissant (mirroir) qui ne concerve aucune trace. Le cas où a_1 vaut -1 est un cas particulier, puisque le système converge en une seule interaction vers sa valeur limite qui est b_1. Il faut ajouter à cela que la valeur maximale atteinte dépend également de la valeur de a_1 (et de b_1). Cette valeur maximale atteindra *IR_MAX*, quelque soit la valeur de b_1, si a_1 est supérieure ou égale à 0 ; en revanche elle sera égale à b_1 pour a_1 égale à -1 et à $(n*b_1)$ pour a_1 inférieure à 0 avec $(a_1=1/n)$. Dans ces deux derniers cas, si b_1 ou $(n*b_1)$ sont inférieurs à *IR_MAX*, cette valeur ne sera jamais atteinte. Nous verrons dans l'exemple de la ré-organisation des clients et des serveurs dans les problèmes d'optimisation de ressources, comment ces propriétés (faire tendre *IR* vers des limites inférieures à la valeur maximale) permettent l'introduction du modèle concentrique.



avec $t_{(Ei)}$ qui correspond au temps où s'est fait l'interaction "Ei".

Dans le cas particulier où a_1 vaut -1, si b_1 vaut *IR_MAX* (cas particulier, cf. le schéma ci-contre) il s'agit d'un système très simplifié pour lequel l'excitation et l'oubli sont quasiment booléen. Un tel cas ne permettra pas d'envisager des constructions de restructuration opérationnelles complexes, comme celle des anneaux concentriques, proposée dans le cadre de l'optimisation du partage des ressources (cf. le chapitre suivant).

Terme B : influence de la forme

$$B_a = \left[\rho_{ndi,\Phi} * k\,(1/m)\right] \qquad \text{cas (a)} \qquad B_b = \left[\frac{M_{ndi}}{M_{BCO}} * k\,(1/m)\right] \qquad \text{cas (b)}$$

$$\rho_{ndi,\Phi} = \frac{\delta M_{ndi,\Phi}}{L_\Phi} \quad \text{avec } L_\Phi \text{ qui est la limite catégorielle de la forme } \Phi$$

$$\lim_{\Phi \to BCO} (\delta MR^{+}ndi,\Phi) = MR^{+}_{ndi} \quad \text{et} \quad \lim_{\Phi \to BCO} (\delta MR^{-}ndi,\Phi) = MR^{-}_{ndi} \quad \Rightarrow \quad \lim_{\Phi \to BCO} (\delta M_{ndi,\Phi}) = M_{ndi} \qquad (i)$$

$$\text{de même,} \quad \lim_{\Phi \to BCO} (L_\Phi) = M_{BCO} \qquad (ii)$$

$$(i) + (ii) \Rightarrow \boxed{\lim_{\Phi \to BCO} B_a = B_b}$$

Ce terme s'intéresse au rôle de la partie du référentiel local du noeud courant, dans la forme où il intervient. Le flux excitatoire est modulé en fonction de l'importance relative du noeud origine dans la forme qui l'englobe. La densité $\rho_{ndi,\Phi}$ représente une densité locale à la forme, centrée sur le noeud nd_i. Un renforcement du flux excitatoire correspond a un noeud important dans la forme. Contrairement au terme *A* centré sur la topologie du référentiel du noeud origine, le terme *B* prend davantage en compte la notion de quantité de masse relationnelle produite dans ce référentiel. C'est une densité donc la dimension relationnelle de la forme compte ; mais pour des formes de dimensions relativement comparables, c'est la masse du noeud origine qui sera discriminant. Pour des formes comparables, plus un noeud possède une masse relationnelle importante et plus les flux excitatoires propres aux liens dont il est l'origine, seront grands. On utilise dans la formule du flux *EXCI'* le carré de *B* pour rendre ce terme moins important que *A*. L'équivalence des termes B_a et B_b se justifie de la même façon que pour l'équivalence des termes A_a et A_b lorsque le noeud étudié n'est pas dans une forme collective.

Terme C : influence de la base

Ce terme s'intéresse au rôle de la forme du noeud courant, dans la base. Il module le flux excitatoire en fonction de l'importance relative de la forme du noeud courant dans la base. Au travers ce terme on impacte l'évolution du flux avec la densité de la forme par rapport à la base. Pour une base dont l'activité relationnelle est assez stable dans le temps, ce terme donnera d'autant plus d'importance à la masse relationnelle (L_Φ) produite par cette forme que cette dernière sera grande. Comme pour le terme *B*, ce troisième terme *C* intègre donc l'heuristique qui augmente l'attraction en fonction de la quantité de masse relationnelle du contexte. Ici il s'agit de la masse de la forme. On utilise le cube de ce terme dans la formule pour diminuer son influence dans le calcul final.

$$C_a = \left[\rho_{\Phi,BCO} * w\,(1/p) \right] \qquad \text{cas (a)}$$

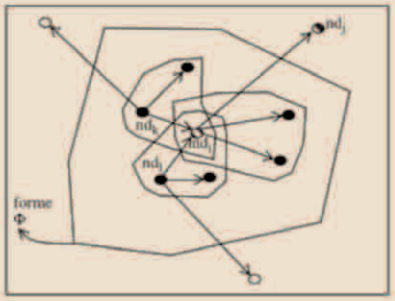
$$C_b = \left[0 * w\,(1/p) \right] = 0 \qquad \text{cas (b)}$$

Schéma-13

Lorsque le noeud étudié n'est pas inscrit dans une forme collective, ce terme est nul puisque dans ce cas la réduction au rapport noeud sur base est donné par le terme précédent *B*. Dans ce cas la masse de la forme ne joue naturellement aucun rôle sur le flux excitatoire, puisque cette forme n'existe pas.

Une approche répartie

Nous revenons sur le calcul du flux excitatoire *EXCI'* en ne considérant cette fois que les deux termes *A* et *B* qui correspondent aux deux niveaux de visibilité disponibles en réparti. Il s'agit d'une part du rôle du lien courant vis-à-vis de l'ensemble des liens vus par le noeud origine dans la forme corrrespondante. On retrouve exactement le terme *A* de l'approche *partiellement centralisée*. Ce terme correspond bien à une certaine description topologique locale du référentiel local représentant le noeud origine du lien courant étudié. Le deuxième niveau de visibilité est celui des noeuds (*ndk* et *ndl* sur le schéma ci-dessous) qui activent directement le noeud origine *ndi* de ce lien. L'évolution du flux excitatoire dépend dans le deuxième terme *B*, de la topologie la plus proche de celle définie dans *A*, c'est-à-dire de celle des noeuds origines des liens arrivant sur le noeud origine *ndi* du lien courant.



Cas (a) : le noeud *ndi* appartient à une forme Φ.

où :

$EXCI'_{ij}$ est le flux excitatoire du lien de nd_j vers nd_i
Φ la forme relationnelle collective dont nd_i fait parti
Voyant(nd_i) est l'ensemble des noeuds du graphe tels qu'il existe un lien depuis chacun de ces noeuds vers nd_i.
Card(ens) est la fonction qui renvoie le nombre d'éléments contenus dans l'ensemble *ens*.

$$EXCI'_{ij}(t) = \frac{1}{2}\left[A * B^2 \right]$$

$$EXCI_{ij} = \frac{1}{2} \times \left[\left(\rho_{ij,\Phi \cap MR^{+}_{ndi}} \times h\left(\frac{1}{n}\right) \right) + \left(\frac{\sum_{s=1}^{v} \rho_{si,\Phi \cap MR^{+}_{ndi}}}{v} \right)^2 \right]$$

avec : n = nombre de liens dont nd_i est origine.
$h()$ est définie de N^{+*} dans $[-1, 1]$

$\forall s,\ nd_s \in \{VOYANT(nd_i) \cap \Phi\}$ et $v = Card(VOYANT(nd_i) \cap \Phi)$

sur le dessin :
nd_l et $nd_k \in \{VOYANT(nd_i) \cap \Phi\}$
$v = Card(VOYANT(nd_i) \cap \Phi) = 2$

Rappel

$$\rho_{ij,\Phi \cap MR^{+}ndi} = \frac{\alpha_{ij} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})}$$

avec $\forall$ p, ndp $\in$ {VU (ndi) $\cap$ Φ}

La présence du carré amoindrit l'influence de ce terme *B* dans le calcul de *EXCI'*. On peut considérer que cette topologie est plus lointaine que celle définie par *A*. Le calcul de ces termes *A* et *B* ne posent aucun problème en réparti. Dans le protocole de mise à jour des caractéristiques évolutionnistes de chaque noeud, une première phase consiste à ce que chaque noeud calcule sa nouvelle densité locale d'attraction ($\rho_{ij,\Phi r \supset MR^+ ndi}$) ; cette première phase peut être faite en parallèle sur toute la base. Une deuxième phase consiste à ce que chaque noeud nd_i diffuse $\rho_{ij,\Phi r \supset MR^+ ndi}$ aux noeuds qu'il active. Une troisième phase permet alors en parallèle sur toute la base, d'effectuer le calcul par chacun des noeuds des nouvelles valeurs de ses flux excitatoire et entropique pour la nouvelle période de fonctionnement à venir.

Cette expression *EXCI'* prend en compte non plus l'influence globale de la forme incluant le lien étudié, mais une partie locale. Le flux d'un lien sera d'autant plus fort que la densité de la partie de la forme touchant ce lien sera grande. Cette influence est d'autant plus grande que la topologie considérée est proche du lien concerné. Là encore suivant le domaine d'application et les besoins associés, l'hypothèse retenue peut être adaptée. L'essentiel est de valider expérimentalement ou par simulation ces hypothèses.

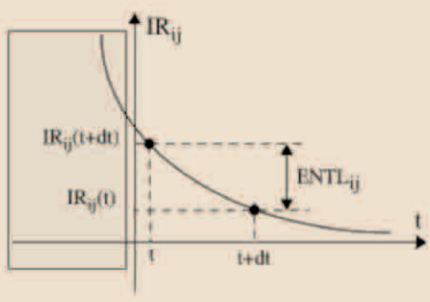
Dans le cas où le noeud *ndi* origine du lien courant, n'appartient pas à une forme, le calcul est étendu à l'ensemble des noeuds cibles de chacun des référentiels locaux concernés. On étend alors l'importance de la topologie contextuelle. Quand cette forme existe, l'influence topologique contextuelle est limitée par cette dernière, afin de donner plus d'importance aux formes dans le modèle d'attraction.

On retrouve pour le terme *A* les mêmes caractéristiques d'influence que celles décrites dans l'approche *partiellement centralisée*. Par contre pour le terme *B* il s'agit d'une moyenne sur les zones limitrophes. Plus cette moyenne correspond à une densité locale faible et moins l'attraction sera augmentée par ce terme *B*. Le carré atténue cette influence.

ii) Entropie

Dans la mesure où l'utilisation du modèle relationnel suit l'heuristique selon laquelle la tendance est de regrouper les entités qui entretiennent entre elles un haut niveau d'interaction, il fallait introduire une composante qui s'oppose à des agglomérations définitives. C'est le rôle que joue le flux entropique en diminuant systématiquement (en fonction du temps) les valeurs des liens en fonction du contexte.

Nous appelons donc entropie la propriété qui s'applique aux liens relationnels de la base et dont le rôle est de réguler les augmentations de ces liens dûes aux activations. Il s'agit d'éviter des phénomènes d'agglomération locale. Pour cela nous définissons la notion de flux entropique *ENTL'*, qui est le "symétrique" du flux excitatoire *EXCI'*. Propre à chaque lien relationnel de la base, il joue continuement sur l'intensité relationnelle *IR* entre les noeuds en diminuant cette valeur dans des proportions liées au contexte relationnel du lien (noeud, forme et base). De cette confrontation entre flux excitatoire et flux entropique, naît une dynamique d'attraction et de répulsion dans l'espace relationnel dont nous présenterons une synthèse dans la partie 3.3.

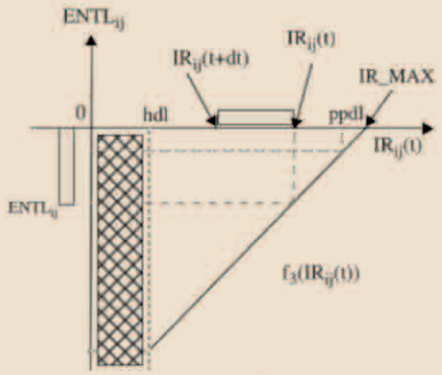


$$IR_{ij}(t) = sup((IR_{ij}(t) + ENTL_{ij} \times dt), IRMIN)$$

Dans (bou-a 92) nous avions proposé que l'entropie (*ENTL*) dépende (f_3) de l'intensité relationnelle (*IR*).

$$ENTL_{ij} = f_3(IR_{ij})$$

Il existe également un flux entropique propre à la base. Il est appelé dans [bou 92] entropie globale *ENT* et joue sur les liens entre la base et chacun de ses noeuds via le coefficient appelé *altitude dérivationnelle AD*. Dans le soucis de simplification déjà mentionné, nous ne traitons pas ce flux dans la présente étude.



Le flux entropique local *ENTL'* joue sur le coefficient α_{ij} attaché à chaque lien nd_i-nd_j de la base en diminuant continuement sa valeur. Si aucune activation n'est plus menée entre deux noeuds préalablement activés entre eux, alors au bout d'un temps fini la valeur *IR* du lien deviendra nulle. Ceci pouvant avoir comme conséquence la rupture du lien pré-existant (cela dépend du choix des options de restructuration du système).

La dynamique introduite par ce flux entropique ne doit pas remettre en cause les heuristiques intégrées dans le calcul du flux excitatoire *EXCI*. Au contraire le calcul du flux entropique renforcera à sa manière l'intégration de ces heuristiques. Le calcul de *ENTL'* se fait suivant les deux mêmes cas que pour le calcul de *EXCI'*.

En choisissant une fonction affine (a_2, b_2) pour f_3, *IR* devient une fonction exponentielle du temps. Avec :

$$f_3(IRij) = a_2 IR_{ij}(t) + b_2$$

on obtient :

$$\boxed{IR_{ij}(t) = k \times e^{a_2 t} - b_2 / a_2}$$

où k est une constante.

Dans (bou-a 92) nous avions appelé "pente entropique (locale)" *ENTL'*, pour le lien entre les noeuds nd_i et nd_j au temps t, la pente de f_3. Nous parlions alors pour cette grandeur de "flux entropique (local)" lié au lien relationnel nd_i-nd_j, caractérisé par IR_{ij}.

Nous avons proposé dans (bou-a 92) de relier cette grandeur (*ENTL'*) à une fonction (f_4) de la masse relationnelle de nd_i (noeud origine de l'interaction). Cela permettait au référentiel local de jouer un rôle dans l'évolution de cette grandeur. En particulier on pouvait ainsi introduire dans le modèle la notion d'attraction relationnelle, avec toutes les heuristiques possibles qui en découlent. Par exemple, plus un noeud joue un rôle relationnel important (nombreuses interactions avec beaucoup d'autres noeuds) plus il a une masse relationnelle importante et moins l'entropie va agir sur ses liens ; ou encore plus le nombre de noeuds avec lesquel un noeud donné est relié, plus l'entropie agira pour "dé-intensifier les liens, ...

$$\boxed{ENTL'_{ij} = \frac{d}{dIR}(ENTL_{ij}) = \frac{d}{dIR} f_3(IR_{ij}) = a_2 \qquad ENTL'_{ij} = f_4(M_i)}$$

Comme pour le flux excitatoire (*EXCI'*), nous introduisons dans f_4 la notion de topologie relationnelle du contexte du lien sur lequel agit le flux entropique (*ENTL'*) et nous nous basons sur les paramètres de densité, précédemment introduits, pour f_4.

Une approche partiellement centralisée

Les topologies et les masses relationnelles contextuelle sont pris en compte symétriquement par rapport au calcul de *EXCI'*. Nous ne détaillerons pas ces calculs.

Cas (a) : le noeud *ndi* appartient à une forme Φ.

où : ENTL'ij est le flux entropique du lien de *ndi* vers *ndj*
Φ la forme relationnelle collective dont *ndi* fait partie
BCO la base étudiée

$$ENTL'_{ij} = f_4 = \frac{1}{3}\left[A + B^2 + C^3\right]$$

$$ENTL'_{ij} = \frac{1}{3}\left[\left(\rho_{ij,\Phi \cap MR^{+}_{ndi}} - 1\right) \times h\left(\frac{1}{n}\right) + \left((\rho_{ndi,\Phi} - 1) \times k\left(\frac{1}{m}\right)\right)^2 + \left((\rho_{\Phi,BCO} - 1) \times w\left(\frac{1}{p}\right)\right)^3\right]$$

avec : n = nombre de liens dont nd_i est origine et dont l'extrémité est un noeud de Φ.

m = nombre de noeuds de Φ

p = nombre de formes de la base

h(), k () et w () sont définies de N^{+*} dans [-1, 1]

Dans le cas suivant, où ndi n'appartient à aucune forme, le terme C est nul.

Cas (b) : le noeud *ndi* n'appartient pas à une forme Φ.

où :
ENTL'ij est le flux entropique du lien de *ndi* vers *ndj*
Φ la forme relationnelle collective dont *ndi* fait partie
BCO la base étudiée

$$ENTL'_{ij} = f_4 = \frac{1}{2}\left[A + B^2\right]$$

$$ENTL'_{ij} = \frac{1}{2} \times \left[\left(\left(\frac{a_{ji} \times MS_{ndj}}{MR^{+}_{ndi}}\right) - 1\right) \times h\left(\frac{1}{n}\right) + \left(\left(\frac{M_{ndi}}{M_{BCO}} - 1\right) \times k\left(\frac{1}{m}\right)\right)^2\right]$$

Terme A : influence du référentiel local

Il s'agit de regarder l'influence de la topologie du référentiel local d'un noeud donné, dans l'évolution des flux entropiques qu'engendre ce référentiel local d'accessibilité. On procède de la même façon qu'avec les flux excitatoires : la stabilisation des formes passe par l'augmentation des écarts en préservant les liens forts et par une limitation d'autant plus grande du resserrement des liens, que les topologies locales sont complexes.

Ici il s'agit de dilatation des liens (diminution de *IR*) ; plus le référentiel local attaché à un lien sera complexe et plus la dilatation de ce lien sera grande.

On retrouve pour les cas rencontrés, la répartition en deux catégories, définie dans la description des flux excitatoires : les équirépartis et les autres.

$$A = \left[\left(\rho_{ij,\Phi \cap MR} + {}_{ndi} - \Gamma \right) * h\,(1/n) \right] = \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} - 1 \right] * h\,(1/n)$$

Avec n, étant égal au nombre de noeuds extrémités des liens partant du noeud courant.

① *Equirépartition des activations depuis nd_i*

$$\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp}) = n * (\alpha_{ji} * MS_{ndj}) \quad \Rightarrow \quad A = ((1/n) - 1) * h\,(1/n)$$

$$\lim_{n \to \infty} A = \lim_{n \to \infty} h\,(1/n) = 0 \qquad \text{si} \quad n = 1 \quad \text{alors} \quad A = 0$$

Commentaire : Dans les deux cas le flux entropique est nul. Cela signifie qu'en cas d'équirépartition l'évolution du flux entropique est constante, d'où une forme de stabilité. Ce qui est intéressant c'est de comparer les flux entropique et excitatoire dans la même circonstance. On voit que pour un seul lien (n=1) le premier terme $A_{EXCI'}$ du flux excitatoire est maximal, alors que $A_{ENTL'}$ est minimal. Dans ce cas on contracte le plus possible un tel lien tout en le dilatant par entropie le moins possible. On favorise donc la stabilité de tels liens.

$$\lim_{n \to \infty} A_{EXCI'} = \lim_{n \to \infty} h\,(1/n^2) = 0 \qquad \text{si } n = 1 \text{ alors } A_{EXCI'} = 1$$

$$\lim_{n \to \infty} A_{ENTL'} = \lim_{n \to \infty} h\,(1/n) = 0 \qquad \text{si } n = 1 \text{ alors } A_{ENTL'} = 0$$

Lorsque le nombre de liens *n* devient grand $A_{EXCI'}$ et $A_{ENTL'}$ tendent vers la valeur nulle mais pas de la même façon (1/n et $1/n^2$). L'excitation est beaucoup plus faible que l'entropie qui reste néanmoins très faible.

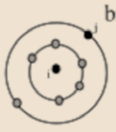
② *Non équirépartition des activations depuis nd_i*

❶ la branche courante (ij) est nettement plus active que l'ensemble des autres branches :

$$\lim \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} - 1 \right] = 0 \quad \Rightarrow \quad \lim A = 0$$

Commentaire : Dans ce cas on a une très faible variation du flux entropique ; ce qui accentue le renforcement de ce lien dû au flux excitatoire.

❷ la branche courante (ij) est nettement moins active que certaines autres branches :



$$\lim \left[\frac{\alpha_{ji} * MS_{ndj}}{\Sigma_{p=1,n}\,(\alpha_{ip} * MS_{ndp})} - 1 \right] = -1 \quad \Rightarrow \quad \lim A = \lim -h\,(1/n)$$

Commentaire : Dans ce cas on a une forte variation du flux entropique ; ceci accentue l'écart entre ce lien et les autres.

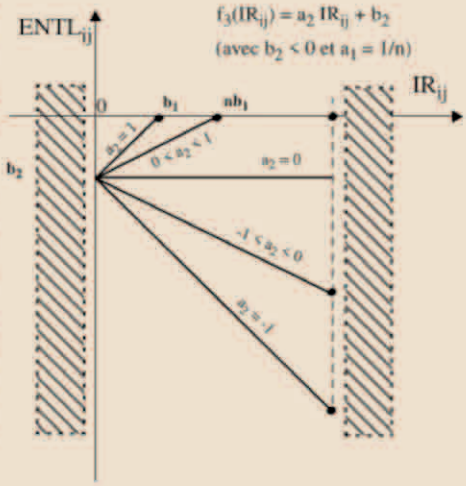
Revenons sur la fonction d'entropie *ENTL* qui diminue continuement dans le temps, la valeur (IR_{ij}) du lien relationnel nd_i-nd_j.

Puisque cette fonction (f_3) est une fonction affine (a_2, b_2) de *IR*, et en faisant abstraction des termes *B* et *C* définis précédemment, sa pente ($a_2 = ENTL'$) doit être comprise entre -1 et 1.

on a l'un des deux cas suivants : $(ENTL_{ij} \leq 0 \quad et \quad a_2 \in [0,1]) \quad ou \quad (ENTL_{ij} \leq 0 \quad et \quad a_2 \in [-1,0])$

Pour simplifier l'étude (cf. la partie identique concernant l'excitation) nous prenons ($b_2 = - b_1$). Nous ferons jouer à l'entropie un rôle presque symétrique de celui joué par l'excitation, dans la mesure où l'entropie agit à chaque intervalle de temps (*dt*) alors que l'excitation n'intervient qu'après chaque interaction.

L'entropie (*ENTL*) est, par définition, une quantité négative ou nulle ; en effet, l'entropie fait décroître continuement dans le temps la valeur des liens relationnels. Ceci implique que ($f_3 = a_2\ IR_{ij} + b_2$) soit toujours négative. Puisque *IR* est toujours positive ou nulle et que nous avons choisi ($b_2 = - b_1$), alors a_2 peut être négatif ou positif (cf. le schéma ci-contre)..



Pour comprendre le rôle de l'entropie on va considérer que *IR* a atteind sa valeur maximale, puis on va regarder comment l'entropie agit dans le temps sur *IR*. On verra dans la partie suivante comment les deux phénomènes se superposent. Néanmoins il faut déjà considérer les champs de valeurs de *IR* en fonction des différents cas prévus par le flux excitatoire. En effet, le réglage de *EXCI* conditionne les valeurs possibles (y compris extrêmes) de *IR*. Nous détaillerons ces cas sur le schéma ci-dessous. En première approximation on peut donner les interprétations suivantes aux cas ci-après :

1) ENTLest nul ($a_2 = 0$)

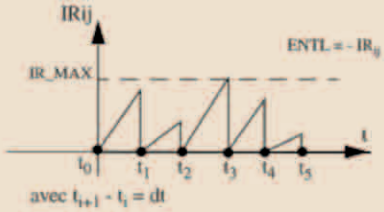
Dans ce cas à chaque *dt*, *IR* diminue de b_2 (variation linéaire). Si cette constante (b_2) vaut 0, alors quelque soit la valeur de *IR*, le flux entropique est sans effet sur le système.

2) ENTLest positif ($0 < a_2 < 1$ et $a_2 = 1$)

L'interprétation de ce cas ($a_2 > 0$) est que plus (cf. le schéma ci-dessous) *IR* est intense (proche de *IRMAX*) et moins l'entropie décroît *IR* ; et inversement (cf. le schéma ci-dessous), moins le lien est intense (faible interaction) plus l'entropie agit sur ce lien. On peut parler d'un système conservateur et froid, car il garde les liens forts et ne favorise pas l'émergence de nouveaux liens. On parlera également dans ce cas de faible élasticité du système.

3) ENTLest négatif (-1 < a < 0)

L'interprétation de ce cas est que plus *IR* est intense (proche de *IR_MAX*) et plus l'entropie diminue l'intensité relationnelle ; inversement, moins le lien est intense (faible interaction) moins ce lien sera affecté (en valeur absolue) par l'entropie. Dans ce cas le système est peu réactif aux interactions.



Un cas particulier est celui où a_2 vaut -1 et b_2 vaut 0. Dans ce cas à chaque *dt* quelque soit la valeur de *IR* et celle de l'excitation (*EXCI*), *IR* passe à la valeur minimale (*IR_MIN* ou 0). En fait on a cette propriété quelque soit la valeur de b_2 et surtout de a_1. Par contre cette valeur a_1 va jouer (cf. le schéma ci-dessous) en théorie pour faire progresser plus ou moins rapidement *IR* vers 0. En théorie seulement, car cela se joue à l'intérieur d'un pas élémentaire *dt* de temps.

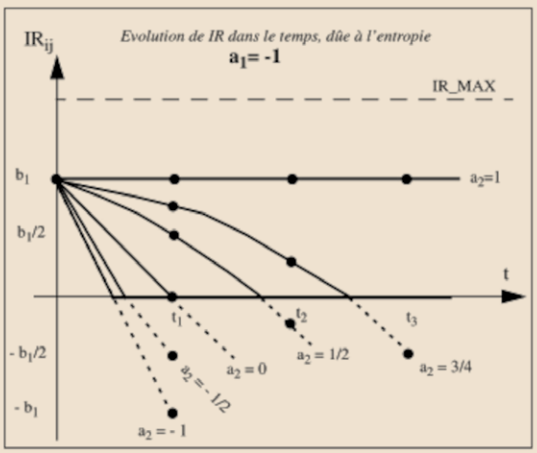
Autrement dit, quelque soit le réglage du flux excitatoire et des interactions subies sur le lien, une seule période de temps *dt* suffit à rendre minimale, voire nulle, l'intensité du lien relationnelle. Un tel système, où l'entropie est maximale, est un système sans mémoire, puisque les interactions sont systématiquement oubliées.

Pour mieux comprendre le rôle de l'entropie sur la variation de l'intensité relationnelle entre deux noeuds, nous allons considérer les trois cas issus des calculs de l'excitation et dont la particularité est de définir trois espaces de valeurs pour *IR*. Ces trois cas sont :

1) $a_1 = -1 \quad \Rightarrow \quad IR_{ij} \in \{0, b_1\}$

2) $-1 < a_1 < 0 \quad \Rightarrow \quad IR_{ij} \in [0, nb_1] \quad avec \quad a_1 = -\frac{1}{n} \quad et \quad n \in N^{+\times}$

3) $a_1 \geq 0 \quad \Rightarrow \quad IR_{ij} \in [0, IRMAX]$



1) Dans ce cas, comme le montre la figure ci-contre, *IR* décroît plus ou moins vite de b_1 à 0 entre les deux situations extrêmes où l'entropie ne joue pas ($a_2=1$) et où elle ramène *IR* à sa valeur minimale en un seul *dt*. On voit également que le système oublie très vite (instantanément) pour toutes les valeurs négatives ou nulle de a_2. Sur le plan théorique, plus on se rapproche de -1 et plus l'on décroit vite (< *dt*).

2) Dans ce cas on observe toujours le même phénomène, avec plus d'ampleur et donc de finesse. En effet *IR* évolue maintenant entre sa valeur minimale (0 sur le schéma) et sa limite nb_1 (calculée précédemment).

Sur le schéma ci-contre nous avons fait figurer quelques asymétries possibles entre *EXCI* et *ENTL*, en particulier au niveau de la valeur de a_1 et de a_2. On voit par exemple que si l'on a :

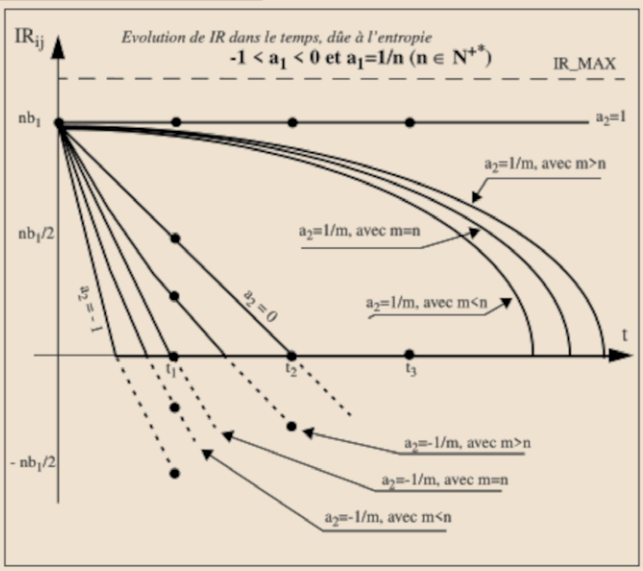
$m > n \quad avec \quad a_2 = \frac{1}{m}$

$et \quad a_1 = -\frac{1}{n}$

$et \quad n, m \in N^{+\times}$

alors *IR* décroît moins vite vers 0 que pour ($m = n$).

3) On retrouve dans ce cas les mêmes comportements qu'au cas précédent. Il faut remplacer la valeur maximale de départ de *IR* qui était nb_1, par *IR_MAX*. Il faut également rajouter pour les cas où

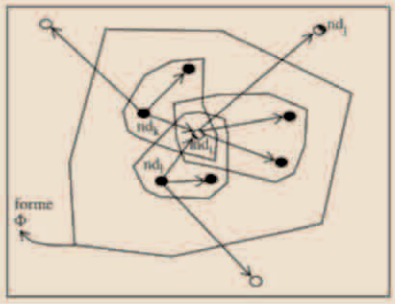


($a_2 = 1/m$), le fait que a_2 doit permettre d'atteindre la valeur *IR_MAX* de *IR*, sinon l'entropie ne joue pas. Cela se traduit par ($a_2 = b_2/IR\_MAX$).

Comme nous l'avons vu en jouant sur les paramètres proposés on peut accentuer ou infléchir le rôle de l'entropie sur le système. C'est le cas par exemple du choix de b_2 vis-à-vis de b_1. Nous avons présenté cette étude avec ($b_2 = - b_1$) ; en prenant par exemple ($- b_2 > b_1$) on renforce le rôle dé-organisateur ou dé-agrégateur de l'entropie vis-à-vis de l'excitation/interaction.

Une approche répartie

Cette approche répartie est le pendant de celle pour le calcul des flux excitatoires. On voit au travers ces formules que pour un modèle d'attraction donné on peut construire un modèle de répulsion (entropie) indépendant ou au contraire symétrique au moins dans l'énoncé des formules. Encore une fois l'expérimentation et la simulation permettront de préciser les choix effectués et les choix possibles.



Cas (a) : le noeud *ndi* appartient à une forme Φ.

où :

ENTL'ij est le flux entropique du lien de *ndi* vers *ndj*
Φ la forme relationnelle collective dont *ndi* fait parti

Voyant(ndi) est l'ensemble des noeuds du graphe tel·s qu'il existe un lien depuis chacun de ces noeuds vers *ndi*.

Card(ens) est la fonction qui renvoie le nombre d'éléments contenus dans l'ensemble *ens*.

$$ENTL'_{ij} = \frac{1}{2}\left[\left(\left(\rho_{ij,\Phi \cap MR_{ndi}} \times h\left(\frac{1}{n}\right)\right) - 1\right) + \left(\frac{\sum_{s=1}^{v}\left(\rho_{si,\Phi \cap MR_{nds}} - 1\right)}{v}\right)^2\right]$$

avec : n = nombre de liens dont nd_i est origine.

h () est définie de N^{+*} dans [-1, 1]

$\forall$ s, $nd_s \in$ {VOYANT (nd_s) $\cap \Phi$} et v = Card (VOYANT (nd_s) $\cap \Phi$)

sur le dessin :

nd_i et $nd_k \in$ {VOYANT (nd_i) $\cap \Phi$}

v = Card (VOYANT (nd_i) $\cap \Phi$) = 2

Rappel

$$\rho_{ij,\Phi \cap MR^{+}ndi} = \frac{\alpha_{ji} * MS_{ndi}}{\Sigma_{p=1,n}(\alpha_{ip} * MS_{ndp})}$$

avec $\forall$ p, ndp $\in$ {VU (ndi) $\cap \Phi$}

L'influence des topologies contextuelles sur le calcul des flux entropiques est principalement centrée sur le nombre de branche (déploiement) des référentiels locaux dans les formes et la densité relationnelle locale.

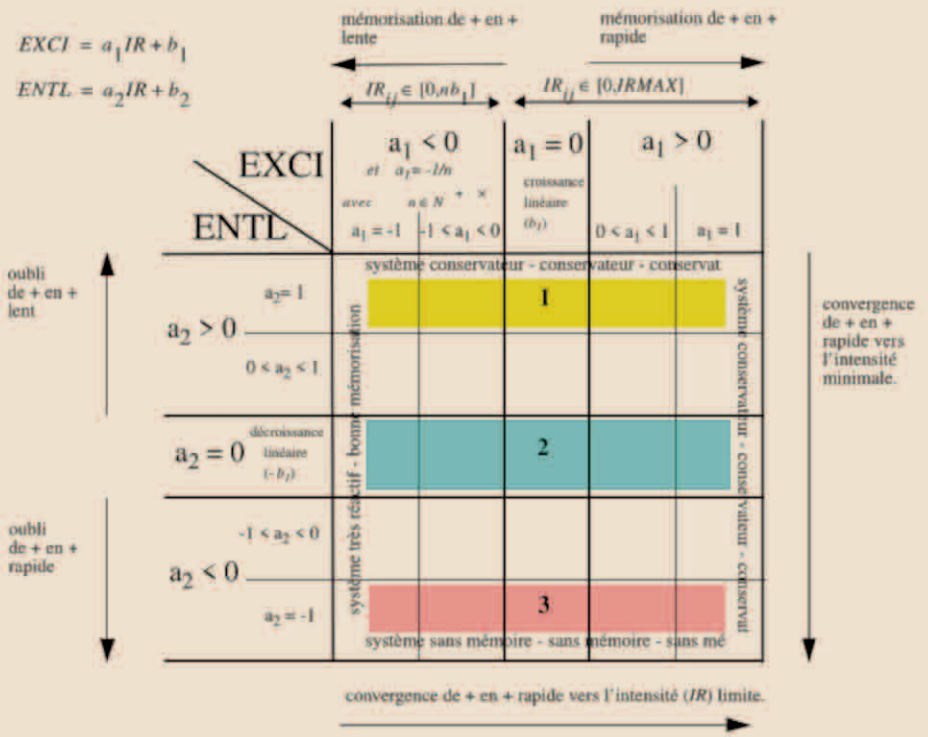
Les calculs de mise à jour de ces paramètres seront faits avec le même protocole que celui décrit brièvement lors du calcul des flux excitatoires dans l'approche répartie.

En conclusion, chaque noeud de la base est donc soumis simultanément à un flux entropique excitatoire et à un flux entropique qui dépendent de sa configuration relationnelle et de celles de son voisinage (forme et base).

iii) Dynamique du système

Nous avons vu qu'en fixant ($b_1 = - b_2$) on permet à l'entropie d'équilibrer l'action de l'excitation dans les mêmes champs de valeurs. Par exemple en choisissant ($a_1 = -1$) et ($a_2 = 1$) on définit le champ de valeurs possibles du système ($0, b_1$). Ceci va être indispensable pour construire des espaces de restructurations exploitables opérationnellement en partant du modèle relationnel. On peut construire d'autres systèmes avec par exemple (b_1 = *IR_MAX*). Dans ce cas le champ de valeurs devient l'intervalle (*0, IR_MAX*) ...

C'est en ce sens que les réglages des flux (*EXCI* et *ENTL*) ne font que modifier des propriétés qualitatives profondes du système, engendrées par la mise en oeuvre de ces flux. Dit autrement, l'application du modèle relationnel permet de renforcer des comportements globaux attendus dans le système étudié. Nous avons déjà présenté les contributions propres à chacun des flux dans la mise en oeuvre d'heuristiques comportementales sur le système. La matrice suivante croise ces propriétés pour dégager les comportements rendus possibles par l'emploi du modèle relationnel dans son ensemble.



Cette matrice fait apparaître trois régions principales qui correspondent à des comportements types du système. Il s'agit de zones dans lesquelles l'entropie joue un rôle particulier (singulier).

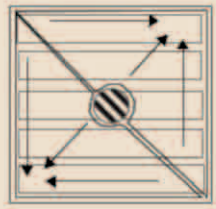
Nous verrons en *3.4* comment les fonctions intégrant les caractéristiques topologiques du contexte relationnel des liens ($h_1, k_1, ...$) , peuvent jouer un rôle important dans le positionnement du système dans cette matrice comportementale.

1) Dans cette région le système est "sans oubli" et ce indépendamment du flux excitatoire. L'entropie n'agit pas sur le système. De tels systèmes risquent des phénomènes d'agrégations irréversibles, les conduisant, d'un point de vue dynamique, vers des états inintéressants.

2) Ici l'entropie agit de façon linéaire dans le temps sur le système. En posant ($IR\_MAX = q \times b_1$) avec q un entier positif, en partant de l'intensité maximale (*IR_MAX*), on atteindra l'intensité minimale en ($q \times dt$). Si en plus on prend ($q = 1$), alors on obtient un système sans mémoire (réfléchissant) qui n'est pas intéressant pour baser des heuristiques de ré-organisation sur justement la trace (mémoire) des comportements antérieurs. En revanche il peut être utile de donner ces caractéristiques à certains composants du système.

3) Enfin cette zone correspond à un système sans mémoire. C'est une généralisation du cas particulier vu en *2)*. En effet, quelque soit l'excitation et *IR*, chaque intervalle de temps *dt*, provoquera un variation de *IR* telle que sa valeur soit égale à la valeur minimale. Même remarque qu'en *2)* sur l'opportunité de tels systèmes.

On peut voir la dynamique du comportement du système d'une autre façon (cf. le schéma ci-contre). Dans ce cas le coin haut droit est le point d'orgue des systèmes à mémoire, alors que le coin bas gauche est celui des systèmes réfléchissant (sans concervation de traces comportementales). Le centre correspond à des comportements linéaires. Un cas particulier est celui pour lequel b_1 vaut 0 et par conséquent b_2 aussi ($b_2 = - b_1$). Dans ce cas les flux (*EXCI* et *ENTL*) n'interviennent pas dans la dynamique du système.



Il faut toutefois insister sur le fait que la paramétrisation des flux (EXCI et ENTL) est propre à chaque lien ndi-ndj ; elle peut donc différer d'un lien à un autre dans le même système. On introduit ainsi une forme deccatégorisation fonctionnelle des objets via la configuration des flux dans lesquels ils évoluent. En donnant des comportements relationnels types (flux) à des groupes d'objets donnés, on peut, comme nous le verrons avec le modèle concentrique dont l'objectif est une discrétisation de l'espace des valeurs des intensités relationnelles, via l'introduction d'attracteurs, introduire dans le système des conditions favorables à l'émergence de comportements collectifs.

Nous allons maintenant présenter brièvement un exemple d'utilisation de ces propriétés à travers ce que nous appelons :

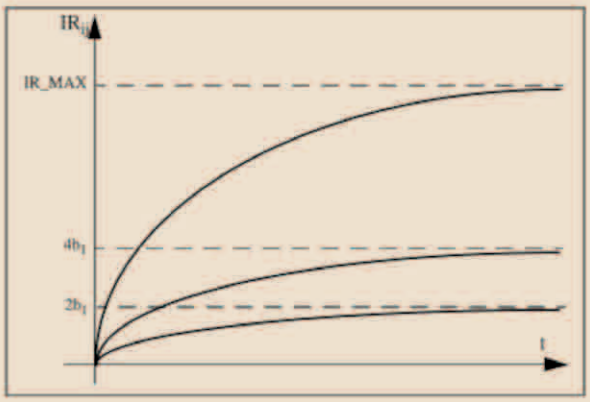
Le modèle concentrique.

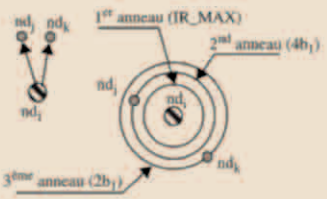
L'objet de ce type de modèle, dont le *modèle concentrique* en est un parmi d'autres, est de proposer un cadre relationnel capable de supporter des stratégies de ré-organisation, ou autres, dans la dimension réelle du système, c'est-à-dire dans sa dimension opératoire (comportements de fonctionnement).

Pour cela nous allons exploiter un espace intéressant de la matrice précédente dans lequel l'espace des valeurs possibles de *IR* est bornée par une limite ($n \times b_1$) qui dépend de *EXCI* (a_1), comprise entre -1 et 0.

Si l'on veut construire un modèle comportemental basé sur une structure concentrique, dans laquelle chaque anneau est un attracteur pour la dynamique de *IR*, il suffit de faire varier a_1 dans l'intervalle ($-1, 0$), et de choisir les autres paramètres des flux de façon ad hoc. Dans ce modèle, chaque noeud (nd_i) du système est au centre d'une structure de ce type. On associe de façon bijective à chaque référentiel local du système, une structure issue du modèle concentrique. Lorsque nd_i interagit avec d'autres noeuds, il voit ces derniers dans son référentiel local, sur une structure de type concentrique. Cela signifie que nd_i est sur le centre de la structure et que les noeuds avec lesquels il interagit sont sur des anneaux (de la structure) concentriques à sa position centrale.

Le comportement recherché est de discrétiser l'évolution de *IR*. Il s'agit en fait de construire des régions dans l'espace des valeurs possibles de *IR* pour lesquelles les flux (*EXCI* et *ENTL*) pourront avoir une paramétrisation propre. Dans notre cas l'espace des valeurs de *IR* est un intervalle (*IR_MIN*, *IR_MAX*), les régions seront des intervalles et des points. La limite de ces régions sera fixée par des valeurs limites de *IR*. On utilisera les propriétés des flux sur la zone de la matrice choisie, qui font converger *IR* vers justement une valeur limite ($n \times b_l$). Cette valeur est alors considérée comme un attracteur pour l'effet du flux excitatoire sur *IR*, dans la mesure où le passage de *IR* vers cette valeur la bloquera dessus.





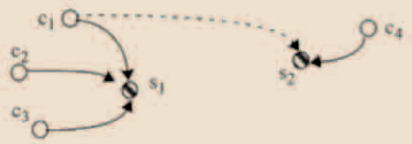
La métaphore du modèle concentrique nous permet de représenter chacune de ces valeurs limites par l'un des anneau concentrique au noeud origine des interactions.

Pour obtenir cette structure il suffit de faire varier les paramètres qui définissent les flux appliqués à un lien donné, en fonction des régions de l'espace des valeurs possibles de *IR*. Sur l'exemple choisi dans les schémas ci-contre, on prend pour *IR* appartenant à $(0, 2b_l)$ a_l égal à -1/2, pour *IR* appartenant à $(2b_l, 4b_l)$ a_l égal à -1/4 et pour *IR* supérieur à $4b_l$, on choisi a_l égal à *-1/IR_MAX*. On voit au passage que l'on peut également régler l'entropie comme on le souhaite région par région.

Le fonctionnement du modèle consiste à faire passer les noeuds extrémités des interactions (nd_j et nd_k) d'un anneau sur un autre, soit vers l'intérieure de la structure (*IR* très intense), soit vers l'extérieure (*IR* très faible). Consécutivement à la première interaction entre deux noeuds donnés, la valeur *IR* du lien conrespondant est de b_l. Les interactions suivantes vont renforcer *IR* pour le conduire vers le centre de la structure, alors que l'entropie diminuera cette valeur dans le temps (*dt*). C'est donc l'entropie qui permettra, consécutivement à la diminution de *IR*, à un noeud de passer d'un anneau donné de la structure à l'anneau directement extérieur. Il en est autrement pour passer d'un anneau donné à l'anneau immédiatement inférieur dans la structure. En effet la valeur limite est définie comme étant un attracteur. Pour cela il faudra définir un seuil, basé par exemple sur une variation ΔIR minimale atteinte et un nombre d'interactions dans cette variation minimale. Si la variation d'intensité relationnelle (ΔIR) est plus petite qu'une grandeur donné depuis un certain nombre d'interactions, alors on autorise *IR* à prendre la valeur lui permettant de

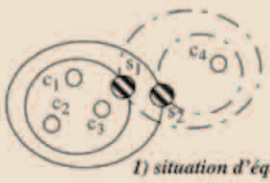
rentrer dans la région suivante (anneau intérieur). La définition de ce seuil mérite sans doute à lui seul une étude assez approfondie ...

Pour illustrer l'utilisation de ce modèle concentrique, prenons le cas suivant, emprunté au problème du partage des ressources dans les systèmes répartis ouverts.



Comme le montre le schéma suivant, la situation relationnelle se compose de 4 clients (c_1, c_2, c_3, c_4) et de 2 serveurs (s_1, s_2) qui rendent un même service ; les trois premiers clients interagissent principalement avec le premier serveur et occasionnellement avec le second, alors que le quatrième client interagit principalement avec le second serveur et occasionnellement avec le premier.

Cette situation correspond à un équilibre relationnel décrit dans le modèle concentrique comme le montre le schéma suivant.

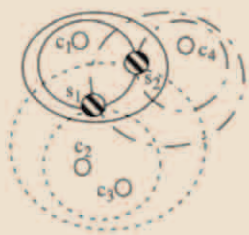


1) situation d'équilibre initial

Dans cette situation les clients c_1, c_2 et c_3 ont des structures relationnelles dans le modèle concentrique, qui se superposent. Pour ces derniers s_1 est sur le premier anneau, alors que s_2 est sur le second. Pour le client c_4, c'est l'inverse. Pour des raisons diverses, comme par exemple une surcharge du serveur s_1 consécutivement à une montée en puissance des demandes de c_2 ou de c_3, il se trouve que le client c_1 interagisse de plus en plus avec s_2, délaissant ainsi s_1. Ceci correspond au lien relationnel en pointillé sur la figure précédente.

Ce changement significatif de comportement de c_1 provoque des variations de IR_{c1s1} et de IR_{c1s2} telles que la situation relationnelle devient la suivante : Pour c_1 il y a inversion de s_1 et de s_2 sur les anneaux de sa structure concentrique. Mais cette inversion peut se faire en deux temps, afin de revenir dans la situation initiale si nécessaire. Pour cela il suffit de jouer sur l'asymétrie des flux. En prenant une entropie *lente* ($a_2 \rightarrow 1$) sur la région délimitée entre le premier anneau et le second, on introduit une latence dans le passage de s_1 du premier anneau vers le second.

Cela nous donne la situation intermédiaire suivante (cf. le schéma ci-contre) dans laquelle c_1 dispose de deux serveurs sur son premier anneau. Nous profitons de cet état transitoire pour montrer qu'il est possible d'introduire sur le modèle concentrique la notion de nombre minimal et nombre maximal d'objet par anneau dans une structure donnée. En terme applicatif cela signifie que l'on veut structurer le réseau avec par client, un serveur attitré (premier anneau) et trois serveurs en secours (deuxième anneau). Là encore cette fonctionnalité mérite un approfondissement de l'étude ... Nous voyons sur cet exemple qu'un serveur peut appartenir à des structures concentriques différentes, c'est-à-dire être simultanément sur des anneaux différents de chacune de ces structures. En revanche il ne peut être simultanément sur des anneaux différents d'une même structure (il possède un seul paramètre *IR* pour un lien donné).



2) situation d'équilibre transitoire

Sur la figure ci-contre nous sommes arrivés à la situation relationnelle qui est conforme aux interactions réelles entre les composants du système, mais également à la structuration/configuration relationnelle recherchée pour l'ensemble du système.

3) situation d'équilibre final

Nous avons discuté le rôle des flux (*EXCI* et *ENTL*) dans le comportement (dynamique) relationnel du système, en fonction des valeurs possibles des paramètres (a_1, b_1, a_2, b_2) qui définissent ces flux. Pour illustrer cette approche, nous allons voir comment elle nous permet de reconstituer l'effet d'interférence et le fan-effect, proposés en psychologie cognitive et relaté dans (cor 87). Pour cela nous revenons sur les calculs de ces paramètres (cf. les formules proposées en *3.1*) dans la mesure où ces calculs intégraient des notions liées à la densité (cf. la partie *2.*) mais également des termes (fonction h_1, k_1 ...) représentant la topologie relationnelle du contexte dans lequel sont définis ces flux.

Pour simplifier, nous prenons uniquement le premier terme proposé dans les formules de calcul de *EXCI'* et *ENTL'* qui ne fait intervenir que le contexte (référentiel) local au noeud origine de l'interaction. Ces paramètres sont en fait le produit entre une densité (ρ) et une fonction (f ou g) de la topologie relationnelle du contexte visé.

$\rho \in [0,1]$ *et* (f, h) *sont définies de* $N^{+\times} \rightarrow [-1,1]$

nous avons : $EXCI'_{ij} = \rho \times f(1/n) = a_1$ *et*

$ENTL'_{ij} = (\rho - 1) \times h(1/n) = a_2$ *avec* $n \in N^{+\times}$

si l'on pose : $f(1/n) = \frac{1}{n}$ *et* $h(1/n) = 1 - \frac{1}{n}$

alors distinguons le cas où n vaut 1, de celui où n est très grand.

1) n = 1, le client ne connaît qu'un seul serveur :

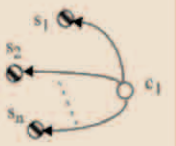
on a $\rho = 1$

$EXCI'_{ij} = f(1/n) = 1/n = 1 = a_1$ *et*

$ENTL'_{ij} = (\rho - 1) \times h(1/n) = 0 = a_2$

dans ce cas on a une décroissance linéaire (- b_1) de IR.

2) $n \rightarrow \infty$ *, le client connaît de nombreux serveurs équivalents :*



on a : $\lim_{n \to \infty} \rho = 0$

$\lim_{n \to \infty} EXCI'_{ij} = \lim_{n \to \infty} [f(1/n) \times \rho] = 0 = a_1$ *et*

$\lim_{n \to \infty} ENTL'_{ij} = \lim_{n \to \infty} [(\rho - 1) \times h(1/n)] = -1 = a_2$

dans ce cas on a une décroissance instantannée (chaque dt) de IR vers sa valeur minimale.

Dans le contexte du partage de ressources dans les systèmes clients/serveurs ouverts, nous pouvons interpréter ces cas de la façon suivante :

1) A chaque interaction entre le client et le serveur, l'accroissement de *IR* suit une exponentielle, alors que l'entropie agit linéairement (- b_1). Le client ne connaissant qu'un seul serveur aura tout intérêt à mémoriser ce serveur (lien) pour pouvoir bénéficier ultérieurement de ses services.

2) En revanche plus *n* est grand et plus les flux conduisent à des liens réfléchissants dfepuis le client vers chacun des serveurs connus. Cela signifie que le client mémorise d'autant moins les serveurs qu'ils sont nombreux. On retrouve l'effet d'interférence où la mémorisation d'un élément est d'autant plus difficile qu'il est composé de nombreux concepts.

Dans l'application aux clients/serveurs, cela peut s'interpréter en disant que plus une population de serveurs est importante, moins les clients doivent s'attacher à l'un d'entre eux *a priori*. Nous verrons au chapitre suivant comment des protocoles de recherche de serveurs basés sur des agents, profiteront de ce type de configuration comportementale.

1.4 . Détection de formes stables

Plasticité et formes collectives

On entend par plasticité d'une forme collective donnée, sa capacité à faire évoluer ses caractéristiques structurelles sans pour autant modifier son identité structurelle. Cette identité structurelle invariante caractérise cette forme. C'est grâce à cette plasticité qu'une forme collective peut évoluer de façon homogène ; et c'est grâce à cette évolution homogène qu'une forme peut se modifier sans remettre en cause sa description invariante (identité structurelle). On voit dans cette description qu'il s'agit de considérer les formes sous l'angle évolutif.

Comme nous l'avons vu précédemment, une forme collective se décrit dans sa totalité ou localement, d'un point de vue théorique (approche catégorielle), et d'un point de veu opérationnel (approche partiellement centralisée ou répartie).

En utilisant les paramètres de densité locale d'un noeud dans une forme et de densité d'une forme dans la base, on travaille sur des rapports qui sont interprétables en terme de rôle d'un noeud dans une forme ou d'une forme dans la base. Ce rôle peut rester le même (à des seuils de tolérance près) même si certaines grandeurs relationnelles comme les masses relationnelles évoluent. Ces paramètres de densité participent à caractériser la structure invariante d'une forme dans une perspective évolutionniste et par conséquent sa plasticité.

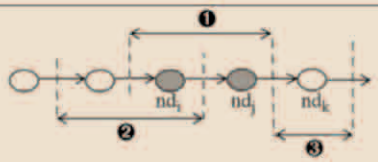
Les interactions entre les individualités font émerger implicitement des formes collectives. L'un des apports de ce modèle relationnel est d'expliciter ces formes collectives pour les intégrer dans le comportement du système afin de mieux faire fonctionner l'ensemble du système. Pour le type de domaine d'application que nous visons, la durée dans le temps d'une forme collective est très importante. En effet toute entité interagissant avec une autre entité produit implicitement une forme collective. Or il faut que cette interaction soit significative, c'est-à-dire qu'elle dure dans le temps, par reproduction du phénomène par exemple. Mais cette interaction peut évoluer au cours du temps et prendre des allures toutes autres ; c'est pour cela que l'on doit observer suffisamment longtemps une forme avant de la prendre en compte en tant que telle ; c'est aussi pour cela que l'on doit être capable de repérer les évolutions des formes sans repartir systématiquement dans un processus de détection de nouvelles formes. Les formes émergent, évoluent et disparaissent.

Stabilité - Homogénéité

Comme nous l'avons déjà dit il s'agit de la stabilité de certains paramètres caractérisant une forme au delà de certaines évolutions de cette forme. Pour reprendre l'image du ballon de baudruche, une forme peut se dilater de façon homogène. Bien qu'elle change progressivement de volume, la forme géométrique reste identifiable. D'autres aspects plus dynamiques peuvent être regardés pour déterminer la stabilité d'une forme ; pour le ballon on peut considérer la vitesse de dilatation qui renseigne non plus sur la simple structure géométrique de notre objet, mais sur les caractéristiques d'élasticité du matériau utilisé. Bien que ces paramètres évoluent, on dira qu'ils le font de façon homogène, c'est-à-dire au moyen de corrélations identifiables.

Propriétés nécessaires au fonctionnement du système

Notre modèle est *réparti* et *asynchrone* (cf. la propagation des excitations et les calculs des flux excitatoire et entropique) ; dans cette optique la notion de forme relationnelle collective est virtuelle ou théorique.



❶ contexte topologique utilisé pour le terme *A* des flux entropique et excitatoire entre nd_i et nd_j ; mais aussi pour la détection des formes.

❷ contexte topologique utilisé pour le terme *B* des flux entropique et excitatoire entre nd_i et nd_j ;

❸ contexte topologique utilisé pour la détection des formes dans les mécanismes d'homogénéisation et d'invariance.

Schéma-14

Dans la pratique les calculs sont faits localement aux référentiels locaux d'accessibilité ; la propagation de ces calculs en suivant les chemins d'excitation, assure l'émergence de formes collectives.

La détection de formes collectives n'échappe pas à ce mode de calculs répartis et par propagation. Nous voyons sur le schéma ci-contre dans quels contextes topologiques se construisent ces calculs et nous rappelons ceux qui permettent les calculs des flux entropique et excitatoire.

C'est dans les mécanismes d'échange entre les noeuds de la base que se construisent dynamiquement les formes collectives. C'est grâce aux mécanismes d'homogénéisation et d'invariance que l'on va détecter les formes et suivre leurs éventuels évolutions.

Détection/caractérisation de formes collectives stables

La détection d'une forme ne se fait pas sur toute la forme d'un seul coup en utilisant un observateur externe à cette forme. En fait localement à chaque référentiel local de la base, des calculs vont permettrent de dire si telle partie de ce référentiel local appartient ou non à une forme. Cette dernière ne sera d'ailleurs jamais référencée en tant que telle.

La détection d'une partie de forme se fait donc toujours à partir d'un noeud nd_i. Cela signifie que l'on ne se préoccupe pas de savoir si les liens éventuels qui arrivent sur ce noeud nd_i appartiennent ou non à une forme. On considère que cela est fait en traitant chacun de ces noeuds comme on traite nd_i.

Dans le domaine d'application du placement d'objets informatiques, cela revient pour un client donné, à savoir si les interactions que ce dernier entretient avec des serveurs sont stables et homogènes dans le temps. Pour cela nous utilisons ce que nous appelons le mécanisme d'invariance perceptive. Cet emprunt de la neurobiologie vient de la métaphore suivante. Dans le système visuel des mammifères, l'invariance perceptive est la faculté que ces derniers ont à pouvoir distinguer le mouvement des objets qu'ils voient dans leur champs de vision dont la cause leur est propre de ceux dont la cause leur échappe. Les mouvements dont ils sont la cause sont dûs au déplacement du mammifère lui-même ou de son système visuel (en tournant la tête par exemple). Les autres mouvements observés proviennent soit des objets eux-mêmes, soit d'une manipulation de ces objets par une entité tierce.

Une métaphore intéressante est la détection par une entité donnée, de la cause d'une modification comportementale ou structurelle d'une autre entité en interaction avec elle. Au lieu de limiter le champ perceptif au champ visuel, on l'élargit au champ interactionnel, via les référentiels locaux d'accessibilité. Le champ moteur couplé au champ visuel dans ces systèmes sensori-moteur, passe des actionneurs de déplacement pour le mammifère aux actionneurs de communication et de restructuration pour la base observée.

L'invariance perceptive.

L'invariance perceptive s'applique à chacun des liens issus de tout noeud nd_i de la base. Il s'agit pour ce noeud de savoir si l'action des interactions qu'il a vers chacun des noeuds qu'il voit, provoque chez le noeud cible une modification structurelle invariante. En d'autres termes quand nd_i active un lien vers un autre noeud, nd_i regarde si l'évolution de ce noeud est corrélée à cette activation.

Sur une période d'observation lorsque nd_i active le noeud nd_j, le protocole de communication impose à nd_j de renvoyer en fin de transaction sa variation de quantité de masse relationnelle $\delta M_{ndj,\Phi}$ dûe entre autre à cette activation. Si nd_j n'appartenait pas à une forme Φ, alors la valeur renvoyée par nd_j à nd_i serait δM_{ndj}. Cette valeur tient compte de l'interaction *ij* mais aussi des propagations issues de nd_j dans son référentiel local d'accessibilité (❸ du schéma précédent) et encore d'éventuelles interactions vers nd_j en provenance d'autres noeuds que nd_i (incluant les cas de formes complexes). Bien que nd_i ne connaisse rien sur le référentiel local de nd_j, si nd_i est capa-

ble de construire une fonction mettant en corrélation les paramètres d'augmentation de quantité de masse relationnelle du noeud nd_i dûe au lien ij et $\delta M_{ndj,\Phi}$, c'est-à-dire la quantité de masse produite par nd_j, alors on considère que le lien entre nd_i et nd_j est inclus dans une forme.

La détection d'une invariance perceptive entre deux noeuds liés, correspond donc à l'existence d'une telle fonction. Chaque noeud regarde lien par lien dans son référentiel local s'il existe de telles fonctions. Dans l'affirmative le noeud marque les liens correspondants comme étant inclus dans une forme. Un exemple simple de fonction d'invariance est le rapport entre les deux paramètres définis ; cela peut être interprété comme une densité locale à la forme Φ, différente des densités précédemment invoquées.

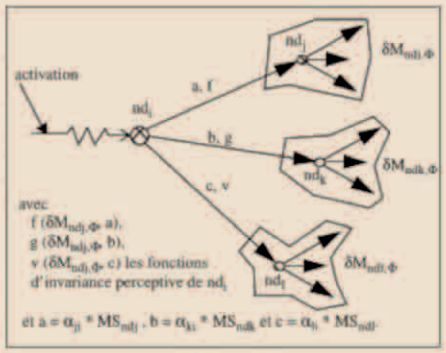


Schéma-15

Le problème ensuite est de savoir si cette forme est *stable*, c'est-à-dire si elle perdure dans le temps discrétisé par les périodes d'observation. Pour cela il faut que les fonctions d'invariance observées, perdurent relativement les unes aux autres. C'est le noeud nd_i, origine du référentiel local correspondant qui va calculer une autre forme de densité locale à son référentiel local. C'est le rapport, pour chacun des liens partant de nd_i, entre la quantité de masse relationnelle (a, b et c) dûe à l'activation de ce lien et l'ensemble de la quantité de masse relationnelle dûe à la partie de la forme étudiée ($a + b + c + \delta M_{ndj,\Phi} + \delta M_{ndk,\Phi} + \delta M_{ndl,\Phi}$). Il s'agit bien d'une densité au sens précédemment introduit, avec des valeurs de sortie comprises entre zéro et un.

Une forme sera donc repérée comme stable, c'est-à-dire repérée tout court, lorsqu'il existera des fonctions d'invariance sur chacun de ses liens et lorsque dans chacun des référentiels locaux traversés par cette forme, les densités locales adhoc seront stables.

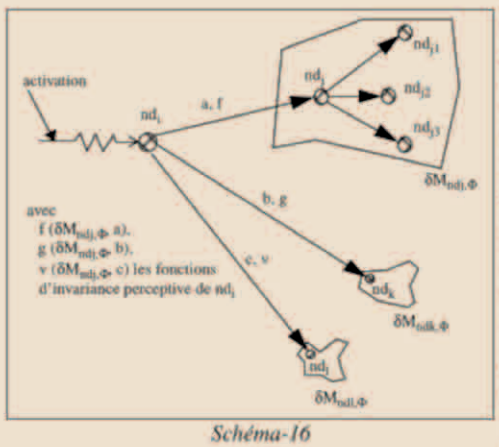


Schéma-16

Ceci complète la définition de la plasticité du système, dans la mesure ou des fonctions d'invariance pourront évoluer sans remettre en cause la forme correspondante, si les densités locales associées restent stables. C'est une façon de modéliser deux fonctionnement locaux différents dans une fonctionalité globale unique.

Sur le schéma ci-contre le noeud nd_j peut utiliser soit le noeud nd_{j1} soit le noeud nd_{j2} ou encore le noeud nd_{j3}. La fonction d'invariance perceptive f intègre une activation moyenne de chacun des trois noeuds. Ceci est vrai pour les autres noeuds nd_k et nd_l. On peut donc associer à la structure invariante d'une forme, tous ces choix correspondant à autant de sous-fonctionalités disponibles dans la forme, ou autant de façon de réaliser la fonctionalité demandée.

Les interactions entre les noeuds évoluant dans le temps au travers des périodes d'observation, il se peut que des formes disparaissent ou évoluent. Soit cette évolution est homogène, c'est-à-dire identique sur l'ensemble de la forme. Dans ce cas les interactions évoluent mais la forme est préservée ; les fonctions d'invariance et/ou les densités locales associées restent identiques. Soit cette évolution n'est pas homogène, alors plusieurs cas sont envisageables.

En fait il faut voir que le modèle de calcul étant réparti, une rupture d'homogénéité est locale à une partie de la forme concernée ; si tant est que cette forme recouvre plusieurs référentiels locaux d'accessibilité. Après avoir détecté cette rupture locale d'homogénéité, il faut voir si la forme de sépare de cette partie non-homogène, si toute la forme évolue ou si des parties de forme passent d'une forme dans une autre.

Navigation dans une forme stable

La navigation dans une forme stable se fait par le marquage des liens. Chaque lien ne pouvant appartenir simultanément à plusieurs formes distinctes, une forme se caractérise par l'ensemble des liens de la base qui sont connexes et marqués, c'est-à-dire disposant d'une fonction d'invariance et d'une densité locale correspondante.

En fait dans l'approche répartie, il n'y aura jamais de navigation dans une forme, cela n'est nécessaire ni pour les calculs des flux entropiques et excitatoire, ni pour le suivi des formes. Il faut considérer la base comme un réseau dans lequel apparaîssent et évoluent en réparti et simultanément des formes collectives.

Répartition des informations liées aux formes collectives

Dans l'approche répartie, les informations liées aux formes collectives se trouvent dans chacun des noeuds propres à ces formes. Les calculs sont effectués en fin de chaque période d'observation. La capture des informations nécessaires à ces calculs est faite continuement.

Dans l'approche partiellement répartie, une partie des calculs des flux entropiques et excitatoire sont traités comme dans l'approche précédente ; pour l'autre partie, les caractéristiques globales aux formes comme leur masse relationnelle (M_Φ) ou leur densité (ρ_Φ) doivent être stockées dans une entité spécifique, c'est-à-dire extérieure à l'un des noeuds du graphe. Cette approche est tout-à-fait réalisable dans le champ d'application du placement des objets informatiques sur un réseau de machines. Les études expérimentales et par voie de simulation permettront de regarder les avantages et les inconvénients des deux approches. Il se peut d'ailleurs qu'une approche fédératrice intègre à la fois les apports de la voie partiellement centralisée avec certains de l'autre voie plutôt basée sur les topologie contextuelles locales.

Détection des ruptures d'homogénéité relationnelle

Les mécanismes de rupture d'homogénéité relationnelle consiste à identifier tous les cas de rupture et à envisager pour chacun d'eux les solutions à mettre en oeuvre.

V-A2) Vers une cosmologie[1] de la connaissance

Les quelques développements que nous allons faire explorent quelques pistes sur la suite éventuelle de travaux à mener vers une représentation des connaissances. Il s'agit davantage d'une tentative de compréhension de ce à quoi pourrait tendre nos objectifs et quelles directions prendre pour les atteindre, ou tout au moins s'en approcher.

Il faut noter qu'un cerveau humain se compose d'environ cent milliards de neurones. Chaque neurone est connecté avec environ dix-mille autres neurones. Nous sommes dans les mêmes ordres de grandeur que le nombre de galaxies dans l'univers. Dans ces conditions il n'est pas complètement déraisonnable d'imaginer un niveau microscopique (l'équivalent de la mécanique quantique) et un niveau macroscopique (l'équivalent de la physique *classique* avec la gravitation dans les corps célestes) dans l'univers des connaissances. Cette dualité, outre le fait de plaider pour un caractère fractal de l'univers qu'il soit physique (la matière, les particules…) ou immatériel (la pensée et les connaissances), ouvre à une lecture cosmologique de l'univers des connaissances.

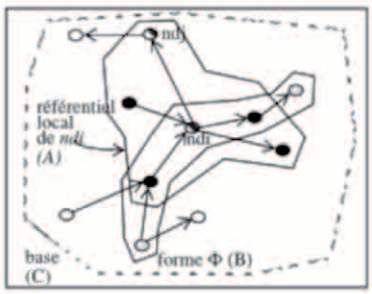


Fig-V-A2-1

Toute cette réflexion, dont la partie émergente est apparue très récemment dans nos travaux, n'en demeure pas moins présente implicitement depuis le début dans la *dérive des connaissances*.

$$M_{Ki,t} = MS_{Ki} + MR^{+}_{Ki,t} + MR^{-}_{Ki,t}$$
Fig-V-A2-2

Comme nous l'avons vu précédemment, et le rappelons sur les figures *Fig-V-A2-1/2/3/4*, la *dérive des connaissances* est un modèle des interactions entre connaissances, dans lequel le temps et l'usager forment un couple moteur et exogène au système de représentation des connaissances. Le modèle, que j'ai proposé, se fonde sur une approche gravitationnelle, au sens de la physique classique, des interactions entre les connaissances de la base de connaissances étudiée.

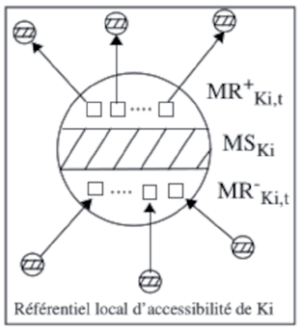


Fig-V-A2-3

Dans mon modèle chaque connaissance dispose d'une masse relationnelle (cf. les figures *Fig-V-A2-2/3*) qui croît avec les interactions entre connaissances, via les usagers, et décroît en fonction du temps. La topologie des liens des connaissances entre elles, joue un rôle (pondération) dans la gravitation interactionnelle (cf. des résultats en psychologie cognitive).

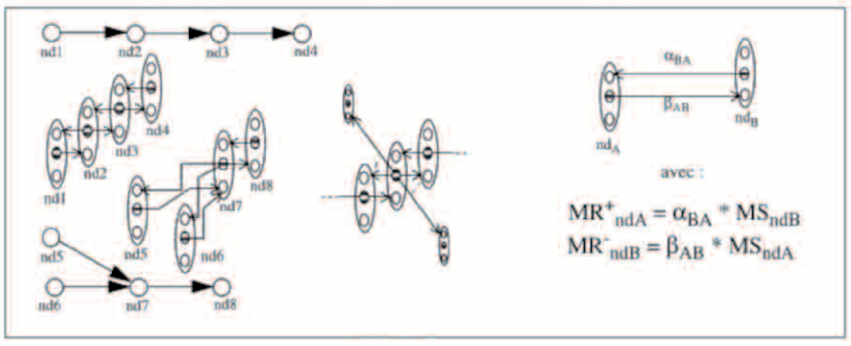


Fig-V-A2-4

Parallèlement, dans la résolution du problème C_nS_k de partage de ressources, j'ai très vite modélisé les demandes des clients non pas comme un flux continu et homogène, mais comme une discrétisation des demandes, par l'intermédiaire des requêtes (quanta). À l'image de la lumière, nous n'avons pas un flux lumineux, mais

1 Les citations en italique de cette partie ont été extraites du livre de Thomas Hertog «L'origine du temps. La dernière théorie de Stephen Hawking», Thomas Hertog aux éditions Odile Jacob, 2023.

une discrétisation de cette dernière en photons. Chaque requête émise (cf. les figures *Fig-V-A2-5/6*), par le flux de demandes d'un client, devient autonome vis-à-vis des autres requêtes du même flux, dans la mesure où elle peut être déviée de sa trajectoire initiale, et donc de celle du flux, en fonction de l'évolution de son environnement.

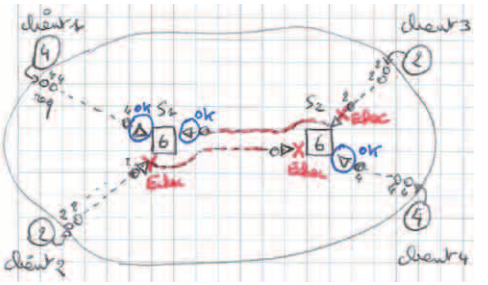


Fig-V-A2-5

Cette autonomie est fondamentale pour expliquer la réaction d'un flux de demandes aux contraintes rencontrées dans l'univers des connaissances. Lorsqu'un flux de demandes (succession de requêtes envoyées par un même client) est émis dans l'univers des connaissances, nos simulations montrent que chaque requête réagit en temps réel (cf. la figure *Fig-V-A2-6*) à l'attraction qu'exercent sur elle les serveurs, par leur simple présence ; ces serveurs étant eux-mêmes en mouvement dans cet univers.

Dans MVB (jusqu'au chapitre IV inclus, c.-à-d. sans recours à la *dérive des connaissances*) le simple fait, pour une requête, de voir un serveur «au plus proche», constitue une forme d'attraction pour cette dernière. Si les serveurs et leurs liens implicites (appelés liens résiduels ou encore interactions résiduelles) forment l'univers des connaissances, même dans MVB cet univers possède une plasticité qui lui autorise des déformations spatiaux-temporelles.

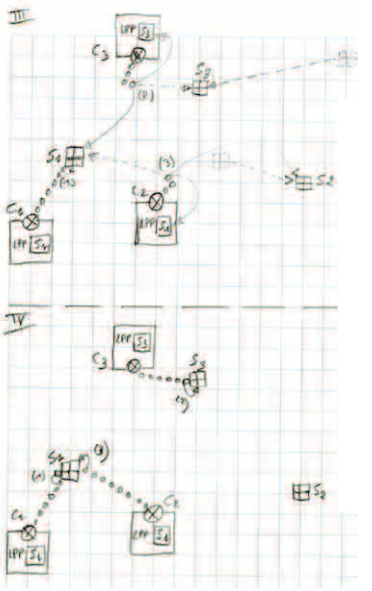


Fig-V-A2-6

Si je réduis l'univers des connaissances ('K') au problème de la résolution du partage de ressources, comme nous l'avons fait dans C_nS_k (aux chapitres III et IV), l'univers de représentation se compose de serveurs avec leur position respective, mais aussi d'une partie cachée correspondant à l'espace des états des solutions. C'est dans cet espace que le système explore d'état en état les différentes solutions au problème posé, en espérant trouver la plus avantageuse pour lui.

p. 330 : «*…Dans la cosmologie moderne apparut, en 1997, l'idée d'une dualité holographique (Juan Maldacena de l'Université Harvard) qui reliait des théories des cordes avec gravitation à des théories de particules sans gravitation. Les deux partenaires, dans la dualité de Maldacena, vivaient dans des espaces-temps à nombre de dimensions différent : la théorie des particules fonctionnait comme un hologramme de la théorie avec gravitation…*
…Maldacena réfléchissait à la théorie des cordes et à la supergravité dans le contexte hypothétique des espaces anti-de Sitter (AdS). Un espace De Sitter permet de décrire un univers s'étendant inexorablement de façon exponentielle (la constante cosmologique d'Einstein 'λ' étant positive). Un espace anti-de Sitter possède une constante cosmologique négative (λ<0) et ne s'étend pas du tout. C'est un container sphérique, sorte de boule à neige, délimité de toutes parts par une surface impénétrable.
L'autre volet de la dualité de Maldacena faisait appel à des

théories quantiques de particules, comme le modèle standard. C'est ce que l'on appelle "théories quantiques des champs", ou TQC, qui décrit les particules et les interactions sous la forme d'excitations locales de champs quantiques présents dans tout l'espace…»

p.331: «*…La surprenante nature holographique de cette dualité vient du fait que les champs quantiques, du côté particule, ne pénètrent pas à l'intérieur de la boule à neige AdS, mais qu'ils demeurent confinés sur la surface qui la délimite. La TQC semble ainsi opérer dans un espace-temps qui possède une dimension de moins. Pour un espace AdS à quatre dimensions d'espace-temps, la TQC évolue en trois dimensions. Il lui manque la profondeur dans l'intérieur de l'AdS, cette dimension courbe perpendiculaire à sa surface. La gravitation est également absente de la TQC: à la frontière de l'espace AdS, il n'y a ni ondes gravitationnelles, ni trous noirs, ni même d'attraction gravitationnelle. La gravitation n'existe simplement pas dans une TQC de particules…*»

p.332: «*…Pour Maldacena, l'holographie rendait équivalente la théorie de la (super)gravité dans l'AdS et la TQC sur la frontière! En conséquence, toute l'information portée par les cordes et la gravitation dans un univers AdS à quatre dimensions peut être encodé dans les interactions quantiques de particules et de champs ordinaires vivant sur sa frontière tridimensionnelle. Le monde de surface serait en quelque sorte un hologramme, une projection du monde intérieur AdS, tout en étant différant profondément en apparence…*
…La dualité holographique établit qu'un monde frontière des champs et particules quantiques, détermine complètement le comportement quantique de la gravitation et de la matière…».

Dualité holographique des connaissances

Cette dualité holographique de la cosmologie peut être transposée dans la représentation des connaissances. Nous parlerons d'un univers AdS-K, comme étant un univers anti-de Sitter adapté à la représentation des connaissances ('K'), et nommerons TQC-K les «théorie quantiques des champs» adaptées à la représentation des connaissances. TQC-K devient la frontière de l'espace AdS-K, et possède trois dimensions. La profondeur vers l'intérieur d'AdS-K, dimension courbe perpendiculaire à la surface TQC-K, est obtenue en introduisant la *dérive des connaissances* dans MVB (→MVB^{++}).

Regardons de plus près cette transposition holographique sur nos travaux présentés autour de MVB.
L'objectif de MVB est de résoudre le problème de partage de ressources, dans un environnement réparti, de façon décentralisée. Pour cela le système cherche la meilleure solution de représentation d'un motif donné, définissant le problème à résoudre. Ce motif est défini par une configuration de clients dans l'espace, qui ont des besoins en ressources dans le temps. La résolution passe par la création de serveurs, qui interagissent entre eux et avec les clients (le motif) via les messagers appelés les requêtes. Dans le cadre de l'étude des lettres de

l'alphabet, chaque lettre (par exemple la lettre 'A') devient un motif.
Le positionnement des serveurs à un instant 't' définit une solution parmi de nombreuses pour atteindre une représentation duale du motif. En clair le problème exprimé sous la forme d'un motif (ensemble de clients disposés dans un espace à deux dimensions et possédant une valeur *demande* exprimée sur la troisième dimension temporelle) trouve une solution sous une forme duale (ensemble de serveurs disposés dans un espace à deux dimensions et possédant une *quantité produite* sur la troisième dimension temporelle).

p.145: «*…Avec la mécanique quantique, le mieux que l'on puisse faire, c'est de prédire les probabilités des différents résultats…*
…En 1925, Schrödinger a établi une équation fascinante qui décrit les particules, non comme de minuscules objects ponctuels, mais comme des entités ondulatoires étendues. Pourtant, et ce point est crucial, les ondes dont il est question dans l'équation de Schrödinger ne sont pas des ondes physiques. En effet, Schrödinger n'a pas prétendu que les particules s'étalaient en quelque sorte dans tout l'espace. Les ondes de la mécanique quantique sont un peu plus abstraites : elles sont plus comme des «ondes de probabilité» qui décrivent les différentes positions possibles des particules ponctuelles. Dans le formalisme de Schrödinger, les endroits où les valeurs de l'onde sont grandes sont ceux où l'on a le plus de chances de trouver la particule. Les endroits où les valeurs de l'onde sont faibles sont ceux où l'on a le moins de chance de trouver la particule…»

p.146: «*…La différence clé avec la mécanique classique de Newton et d'Einstein est que la théorie quantique ne prédit que les probabilités d'états ultérieurs et aucune certitude… demander où se trouve une particule (sa position) n'a même aucun sens. Elle ne possède pas de position définie, seulement des positions latentes décrites par une onde de probabilité qui encode la vraissemblance que la particule, si on la regarde, se trouve ici ou là. Tout se passe comme si, par notre regard, on imposait aux particules d'adopter une position, comme si la réalité tangible n'existait que dans la mesure où nous interagissons avec le monde par l'observation et l'expérimentation. "Pas de question, pas de réponse!", s'exclama un jour Wheeler…*»

L'usage de la *dérive des connaissances*, dans l'espace duale de représentation des motifs, fait directement écho à cette propriété de la mécanique quantique qui consiste à observer une entité pour qu'elle prenne, en quelque sorte, une décision et fixe, par exemple, sa position. En effet c'est l'interaction, exogène (par l'usager) à la base de connaissances, qui détermine les positions des connaissances dans cette dernière. Nous reviendrons ultérieurement sur cette propriété.

p.148 : « *…En décrivant les électrons, non comme des particules qui se déplacent, mais comme des ondes de probabilité qui se propagent, l'équation de Schrödinger prédit que, tout comme les vagues interférant à la surface d'un étang, les divers fragments de la fonction d'onde de l'électron issus des deux fentes*[2] *se superposent et interfèrent, créant un schéma de probabilités hautes et basses concernant l'endroit où chaque électron atteindra l'écran. Là où les fragments d'ondes sortant des deux fentes arrivent en phase, ils se renforcent ; là où ils arrivent déphasés, ils s'annulent. Quand on tire particule après particule, l'accumulation de leurs positions d'arrivée suit le profil probabiliste encodé dans la fonction d'onde individuelle de chaque particule, créant ainsi la structure d'interférences observée…* »

p.149 : « *…L'idée de Feynman fut d'imaginer que les particules sont bien des objets localisés, mais qu'elles suivent tous les chemins possibles lorsqu'elles de déplacent d'un point à un autre. La mécanique classique suppose que les objets empruntent un seul chemin dans l'espace-temps. Par conséquent, un système classique a une histoire unique et bien définie. La mécanique quantique, argumentait Feynman, adopte une vision plus étendue de l'histoire et accepte que tous les chemins possibles soient empruntés simultanément, bien que certains soient plus probables que d'autres… Elle ne prédit que la probabilité d'arriver en 'B' (en partant de 'A'), qui est la moyenne pondérée de tous les chemins reliant 'A' à 'B'…* »

À l'image des chemins quantiques de Feynman, en réalité un motif (problème à résoudre ou à représenter) est décrit par un très grand nombre de solutions possibles, appelées états dans l'espace de représentation. Ces états sont reliés entre eux par un graphe d'état (cf. la figure *Fig-V-A2-7*), un par niveau de profondeur ('k') de la résolution, matérialisant tous les chemins possibles (d'état en état) permettant de relier le motif initial à son état optimal (CG*). Un chemin/trajectoire, dans l'espace de représentation des connaissances, peut être considéré comme une onde permettant de relier/définir, avec une «certaine» probabilité décrite dans la fonction d'onde (à préciser), le motif initial à son état optimal (CG*). À l'image de l'adage populaire «ce n'est pas là où l'on va qui compte, mais le voyage pour y aller»,la représentation du motif dans l'espace dual de représentation des connaissances, n'est pas uniquement l'état optimal (CG*), mais les chemins qui y conduisent.

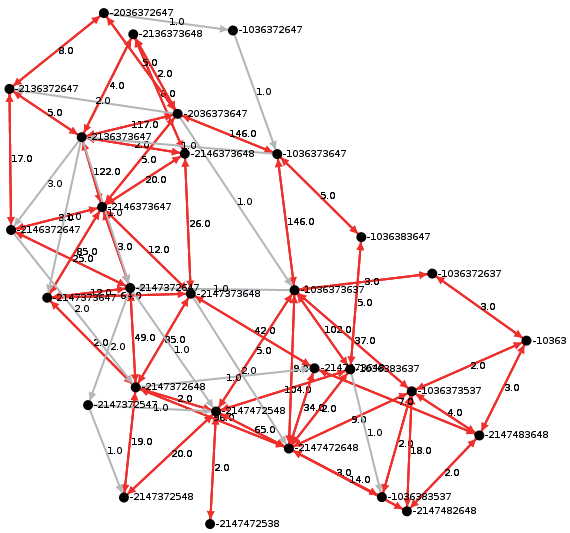


Fig-V-A2-7

L'aspect quantique (TQC-K) revient à déterminer, à partir du graphe exploré, une onde par chemin menant du motif à son état optimal, avec ses grandeurs (probabilités) associées. Dans ce contexte on peut définir la probabilité pour aller du motif à son état optimal (CG*) associé (pour une profondeur 'k' donnée) en additionnant les ondes corrélées à toutes

2 Il est fait référence ici à la célèbre expérience des fentes de Young, réalisée en 1927 aux Bell Labs.

les trajectoires/chemins possibles (moyenne pondérée). Il faut apporter ici une nuance, dans la mesure où «possibles» signifie réellement explorées.
Avec cette vision quantique de l'espace dual des représentations de connaissances, un motif quelconque (son pendant quantique serait donc le versant corpusculaire d'une particule) se définit, non pas simplement par ses 'k' (profondeur d'exploration) états optimaux (CG^*_k), comme s'en suffit la résolution C_nS_k, mais par tous les chemins qu'il peut prendre simultanément pour aller vers ces états optimaux.
C'est la différence essentielle entre l'objectif de C_nS_k et celui de son extension à la représentation des connaissances, qui permettra de mettre en place les mécanismes qui lui sont indispensables.
Dans nos expérimentations nous avons documenté, sur des problèmes/motifs simples, l'ensemble des solutions/états possibles et leurs graphes d'états associés.

Nous avons vu, au chapitre III, que suivant le nombre de serveurs ('k') dont dispose le système dans l'espace dual, dépendront des solutions spécifiques, avec leur graphe d'états associé. Sur la figure *Fig-V-A2-8* nous avons un motif à six points. Le nombre de serveurs varie entre un et six. Sur la courbe de cette figure *Fig-V-A2-8*, apparaît la valeur CG^* (coût optimal) pour chaque graphe d'états associé à chacune des six profondeurs ('k') de résolution du problème initial.

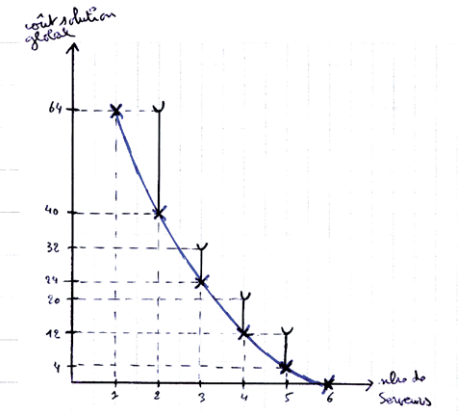


Fig-V-A2-8

Propriété: dans MVB, en utilisant l'algorithme de contraction/expansion (C/E), un motif donné est équivalent, dans la représentation duale, à la somme des graphes d'états associés pour k∈[1,n], où 'n' est le nombre de points (clients) définissant le motif.

Le principe de MVB est d'explorer, autant que nécessaire, l'espace des états, d'un problème (C_nS_k) donné (vu comme un motif unique), pour construire les chemins qui mènent à l'état recherché (CG^*). La complexité du problème/motif nécessite de nombreuses explorations, dans un processus qui peut s'avérer très long. Chaque nouvelle exploration du motif enrichit le graphe des états correspondants.
Le passage de la résolution de type C_nS_k vers la construction d'un système de base de connaissances, nécessite de représenter des connaissances (motifs) en vue d'instancier des modèles (les instances de la lettre 'A', par exemple), de les différencier (entre instances de modèles distincts, par exemple entre 'O' et 'Q') et de les composer entre elles vers des niveaux de représentations supérieurs (par exemple pour former des mots).

Autant avec C_nS_k il suffit de trouver le graphe d'état permettant d'accéder à l'état optimal, autant, pour la représentation des connaissances, il faut considérer chaque motif dans son espace dual, comme étant une particule capable de prendre simultanément tous les chemins qui mènent à ses 'k' états optimaux.

MVB^{++} ET LA *DÉRIVE DES CONNAISSANCES*

L'introduction de la *dérive des connaissances* dans MVB, que nous nommons MVB^{++}, devrait permettre d'apporter la dimension supplémentaire (AdS-K) à la résolution, qui permettra, en introduisant la gravitation informationnelle entre les états et les graphes d'états, la mise en place des fameux mécanismes d'instanciation, de différenciation et de composition, indispensables à la représentation des connaissances.

p.336 : «*…L'ensemble des excitations corpusculaires et des interactions décrites par une théorie quantique des champs (TQC) donnée dépend de la résolution spatiale à laquelle on se place…*»

«*Or cette description varie en fonction de la courbure de l'espace AdS. Cette courbure (associée à la gravitation) se mesure sur l'axe perpendiculaire à la solution 'S' dans TQC. L'origine de cet axe signifie une très faible courbure d'AdS, et plus on s'enfonce dans AdS sur cet axe, et plus la gravitation et le courbure augmentent.*
Le monde frontière quantique (TQC) est dépourvu de gravitation»

p.333 : «*…Tout comme pour d'autres dualités de la théorie 'M', la nature des relations entre les deux partenaires de la dualité holographique est telle que des calculs parfaitement faisables pour l'un peuvent devenir extraordinairement compliqués pour l'autre. Lorsque la gravitation est faible et que l'univers AdS est modérément courbé, par exemple, la description de frontière fait intervenir des interactions quantiques tellement intenses entre ses constituants que la TQC devient totalement inexploitable, et que même la notion de particule individuelle peut cesser d'être pertinente…*»

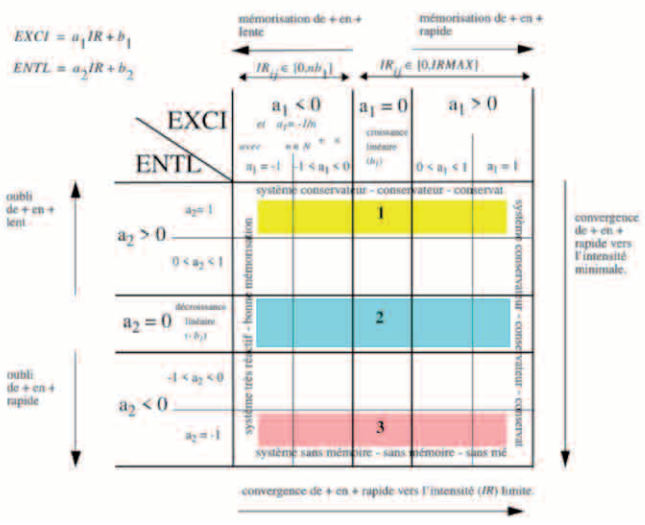


Fig-V-A2-9

Dans la *dérive des connaissances*, les paramètres a_1 (pour les interactions entre connaissances) et a_2 (pour l'effet entropique du temps), décrits sur la figure *Fig-V-A2-9*, permettent la configuration de la dynamique du système de *dérive des connaissances*. Ils jouent le rôle de l'axe perpendiculaire à TQC-K dans AdS-K. En effet suivant leur valeur respective, le système de représentation de connaissances fera apparaître ou non la gravitation informationnelle avec une intensité plus ou moins grande. Par exemple pour $a_2<0$ et $a_1<0$ la gravitation sera trop faible pour jouer un rôle, c'est l'équivalent de l'origine de l'axe AdS où la courbure est très faible. Avec des valeurs positives pour ces deux grandeurs, la courbure d'AdS-K deviendra de plus en plus grande et la gravitation aussi.

La fonction d'onde associée à chaque connaissance de la base décrit les probabilités associées aux différents chemins permettant d'atteindre les 'k' états optimaux (CG^*_k), un chemin étant une onde de la fonction.
Suivant les explorations réalisées, les graphes d'états correspondants peuvent devenir très grands et contenir des informations peu pertinentes. Nous avons vu dans les chapitres III et IV que la complexité du nombre des états et des liens dans les graphes associés à l'étude d'un motif donné, dépend de la richesse du motif (nombre de points 'n', topologie…). Cette complexité est potentiellement très grande, mais le sous-ensemble des états et des liens de ces graphes, porteurs d'informations «utiles», l'est beaucoup moins.

p.152: «*…En ayant, non pas une, mais plusieurs histoires qui se déroulent simultanément, la mécanique quantique limite de façon évidente ce que l'on peut dire du passé. Le passé quantique est intrinsèquement flou. Il ne correspond pas à l'histoire nette et précise à laquelle nous pensons ordinairement quand nous pensons au passé… De façon remarquable, ce schéma de "sommation de toutes les histoires" proposé par Feynman fournit un cadre intellectuel fonctionnel et précis pour penser la théorie quantique en général. Il a été popularisé sous le nom de formulation de la théorie quantique par "intégrale de chemins". Dans l'esprit de Feynman, le monde est un peu comme une tapisserie médiévale flamande - une trame tissée de chemins entrecroisés qui composent ensemble une image cohérente de la réalité, à partir des fils d'une myriade de possibilités…*»

La réalité apparaît comme issue de l'émergence d'une entité supérieure dûe à la superposition de tous les [chemins] possibles.
On peut ainsi lire un modèle (par exemple la lettre 'A' de l'alphabet), au sens de la représentation des connaissances, comme l'émergence quantique dont chaque instance représente l'un des chemins possibles parmi ceux qui mènent à cette réalité.

p.152-153: «*…En effet, la formulation par intégrale de chemin s'applique aussi bien aux objets petits que grands à la différence que, dans le cas de très gros objets, les probabilités se concentrent sur les seules trajectoires proches du chemin unique prédit par les lois newtoniennes classiques du mouvement. Il n'y a donc pas de dichotomie fondamentale entre les mondes micro- et macroscopique. Simplement, pour les objets macroscopiques, les fluctuations microscopiques se moyennent en un résultat défini et déterministe, et ce résultat est le chemin du mouvement classique. Le déterminisme classique, de fait, "émerge" du comportement collectif des histoires quantiques microscopiques aléatoires. À l'opposé, à mesure que l'on s'aventure dans le monde microscopique, il faut prendre en compte une part de plus en plus grande de cet enchevêtrement aléatoire…*»

p.196: «*…La fonction d'onde d'une particule en mécanique quantique fait intervenir un amalgame de tous les chemins possibles suivi par la particule; de même, en cosmologie*

quantique, la fonction d'onde de l'univers décrit l'ensemble de toutes les histoires possibles d'expansion…»

Si je transpose la cosmologie quantique sur les modèles, en représentation de connaissances (par exemple les lettres de l'alphabet vues comme autant de singularités), chaque modèle (lettre) est défini par une fonction d'onde qui décrit toutes les instanciations possibles (passées, présentes et à venir) du modèle.
Il n'y a aucun temps entre ces différentes histoires (instanciations). En fait toute nouvelle histoire enrichit (le fait évoluer) ou non le modèle et se trouve ainsi noyée dans ce dernier.

p.336: «*…la dualité de Maldacena traduit la "profondeur interne" dans l'AdS en "taille" sur la frontière. La première entrée du dictionnaire AdS-TQC expliquerait donc que rétrécir ou grossir dans le monde frontière correspond, dans la partie gravitation, à se déplacer selon la direction perpendiculaire à la frontière, respectivement en se rapprochant ou en s'éloignant du bord.*
De fait, dans les théories quantiques des champs, l'idée d'associer un changement d'échelle à un déplacement dans une dimension supplémentaire remonte à loin. On le sait la taille est étroitement liée à l'énergie. La raison pour laquelle les physiciens des particules requièrent des accélérateurs toujours plus grands est qu'un accroissement de l'énergie des collisions de particules permet de sonder la nature à des distances toujours plus courtes. Accroître l'énergie revient à s'offrir un meilleur microscope. Or, et c'est là un point crucial, l'ensemble des excitations corpusculaires et des interactions décrites par une TQC donnée dépend de la résolution spatiale à laquelle on se place…»

Dans ce contexte, la *dérive des connaissances* a deux principales utilités:

- elle va réduire la dimension (en nombre d'états et de liens conservés) des 'k' graphes d'états, afin de ne garder que les chemins «utiles» (les plus probables) pour réaliser les traitements cognitifs adaptés. En choisissant des valeurs positives pour les constantes a_1 et a_2 de la *dérive des connaissances* (cf. la figure *Fig-V-A2-9)*, nous irons vers une représentation plus «classique» des connaissances, au sens de la physique classique, et non «quantique», avec un ou peu de chemins par graphe d'états, dont la somme des probabilités associées, pour mener à l'état optimal (CG^*_k), tenderait vers 1. Nous allons voir qu'il n'est pas nécessairement intéressant de ne garder qu'un seul chemin pour aller vers l'état optimal E^*. Une vision quantique «allégée» (élimination des chemins «inutiles») permettra de garder de la finesse dans la représentation des connaissances pour faire de la différenciation de modèles;
- elle va aussi gérer la représentation des connaissances dans la base. Son rôle sera d'introduire une courbure dans l'espace de représentation des connaissances (AdS-K) modelant ainsi cet espace en utilisant la

gravitation informationnelle. Plus on se rapproche du modèle associé à une classe de motifs (par exemple toutes les instances écrites de la lettre 'A'), plus la courbure d'AdS-K et sa gravitation informationnelle associée fonctionnent, et moins l'aléatoire entre en jeu dans les mécanismes cognitifs associés. On opère également un changement d'échelle en allant de la représentation microscopique, pour les instances, vers une représentation macroscopique pour les modèles; ce changement d'échelle, dans l'espace de représentation des connaissances autorise la complexification des connaissances manipulées. La courbure de l'espace de représentation des connaissances se base sur les masses relationnelles des différentes connaissances représentées.

Détaillons un peu ces mécanismes. Le premier apport de la *dérive des connaissances* est de réduire, pour un motif donné, les graphes représentant les différents chemins pour aller aux états optimaux (E^*_k) de ce motif.

méta-graphe d'états

Définition: On appelle méta-graphe d'états, MG(motif,k), pour un motif donné (*motif*), l'ensemble des 'k' graphes d'états G_k(motif), qui représentent les différents chemins pour aller vers les états optimaux E^*_k, dans chaque profondeur d'exploration 'k', avec k∈[1,n], où 'n' est le nombre de points (clients) définissant le motif.

Comme le montre la figure *Fig-V-A2-10*, l'exploration dans MVB propose un graphe dont le nombre d'états et de liens entre eux, dépend d'une part d'aléatoire (positionnement initial des serveurs et recours au déplacement aléatoire dans le cycle C/E-G-A), mais aussi du nombre d'explorations effectuées.

La version MVB^{++} (toujours sur la figure *Fig-V-A2-10*), avec la *dérive des connaissances*, produit, pour un motif donné, le méta-graphe MG(motif,k)', dont chaque graphe G_k(motif)' est un sous-graphe «utile» du graphe G_k(motif) correspondant dans MVB.

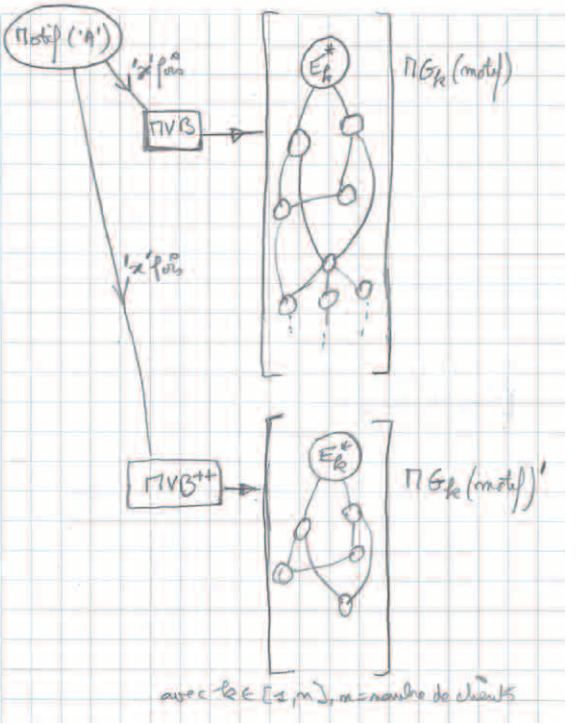


Fig-V-A2-10

sous-graphe «utile»

Définition: On appelle sous-graphe «utile» G_k(motif)' du graphe G_k(motif), obtenu par des explorations successives dans le système de représentation des connaissances, le sous-ensemble des états et des liens associés à G_k(motif) qui détermine les différents chemins pertinants, pour aller vers l'état optimal (E^*_k) associé au motif, dans le contexte d'une différenciation de modèles.

Cette notion de sous-graphe «utile» permet de faire un réglage de précision, à l'image d'une mise au point par

un microscope/télescope, sur la quantité d'information à garder pour différencier les modèles entre eux, mais aussi d'avoir de la latitude pour construire les instances d'un modèle donné.

En effet augmenter la quantité d'informations (le nombre de chemins) pour aller vers l'état optimum permet :

- d'un côté d'ouvrir le champ des possibles de l'instanciation, jusqu'à un certain point au-delà duquel ce champ va, au contraire, se restreindre ;
- d'un autre côté, de fermer ce champ des possibles de l'instanciation. Trop de chemins ferme la variation vers des instances associées en sur-spécifiant le modèle.

Nous savons qu'un modèle (par exemple la lettre 'A'), en représentation de connaissances, n'a de sens que pour être composé avec d'autres modèles (les lettres de l'alphabet), afin de produire des connaissances d'un niveau supérieur (par exemple les mots).
Se pose alors, dans la représentation des connaissances, le problème de la différenciation de modèles. La *dérive des connaissances*, appliquée aux méta-graphes, va jouer un rôle décisif dans cette différenciation. En effet la valeur de 'k' pour un méta-graphe MG(motif,k) va être définie en fonction justement de la différenciation du modèle courant, vis-à-vis des autres modèles du système de niveau supérieur (l'alphabet dans le cadre des lettres). Dit autrement, un modèle donné (par exemple la lettre 'A') aura la plus petite valeur de 'k' comprise entre 1 et 'n' qui lui permettra de se différencier de tous les autres modèles. Par exemple, dans l'alphabet latin majuscule, le système de représentation des connaissances associera une valeur au 'k' de la lettre 'Q' qui lui permette de se distinguer (en terme de chemins, MG(motif,k)…) des autres modèles (lettres), en particulier celles qui pourraient avoir une topologie proche (la lettre 'O'…).

Cela permet de pointer ici le fait que 'n' représente le nombre de points utilisés pour proposer le motif au système de représentation. Cela signifie que chaque motif possède potentiellement un nombre 'n' qui lui est propre. La proposition du motif 'O' ne nécessitera pas autant de point que pour le motif 'Q'. En conséquence chaque motif aura sa valeur 'k', sur un intervalle qui lui sera propre, mais qui dépendra des autres motifs.

Les notions de méta-graphe d'états et ses sous-graphes «utiles» associés, montrent qu'instanciation et différenciation sont des mécanismes liés dans la représentation des connaissances. Un bon système de repésentation devra jouer simultanément dans deux directions, à savoir la profondeur de chaque sous-graphe (nombre de chemins retenus) et la profondeur du méta-graphe associé (nombre de sous-graphes retenus). Ces deux quantités contribuent simultanément à la réalisation d'instanciations efficaces tout en garantissant des différenciations pertinentes.

Chaque graphe G_k(motif)' contient donc l'ensemble des chemins permettant d'aller du motif à son état optimal (E^*_k) dans

l'espace des représentations de connaissances. En nous appuyant sur la figure *Fig-V-A2-11*, nous allons introduire ici les notions de «chemin effectif» et de «chemin réel», qui sont fondamentales dans la représentation des connaissances.

chemins effectifs

chemins réels

Définition: on appele «chemins effectifs», dans un graphe «utile» G_k(motif)', l'ensemble des chemins (successions d'états allant à E^*_k) traversés pendant l'exploration. On appelle «chemins réels» dans un graphe «utile» l'ensemble des chemins possibles à partir de ce graphe G_k(motif)'.
La différence entre les deux provient du fait que des chemins non explorés peuvent néanmoins être déduits du graphe retenu.

Sur la figure *Fig-V-A2-11*, le graphe G_k(motif)' se compose de cinq états dont l'optimum (E^*_k) : $\{E_{1,k}, E_{2,k}, E_{3,k}, E_{4,k}, E^*_{0,k}\}$.
L'exploration a permis de mettre en valeur les chemins effectifs Ce_1 ($E_{1,k} \rightarrow E_{2,k} \rightarrow E^*_{0,k}$) et Ce_2 ($E_{3,k} \rightarrow E_{2,k} \rightarrow E_{4,k} \rightarrow E^*_{0,k}$).
En exploitant le graphe on peut faire apparaître deux chemins réels supplémentaires Cr_1 ($E_{3,k} \rightarrow E_{2,k} \rightarrow E^*_{0,k}$) et Cr_2 ($E_{1,k} \rightarrow E_{2,k} \rightarrow E_{4,k} \rightarrow E^*_{0,k}$) qui n'ont pas été parcourus lors de l'exploration, mais qui existent, bel et bien, parmi les chemins possibles. Ce sont deux chemins qui mixtent les deux chemins effectifs /parcourus Ce_1 et Ce_2.
Le motif choisi, est représenté dans la base de connaissances en intégrant tous les chemins réels ($Cr_1, Cr_2, Cr_3=Ce_1, Cr_4=Ce_2$).

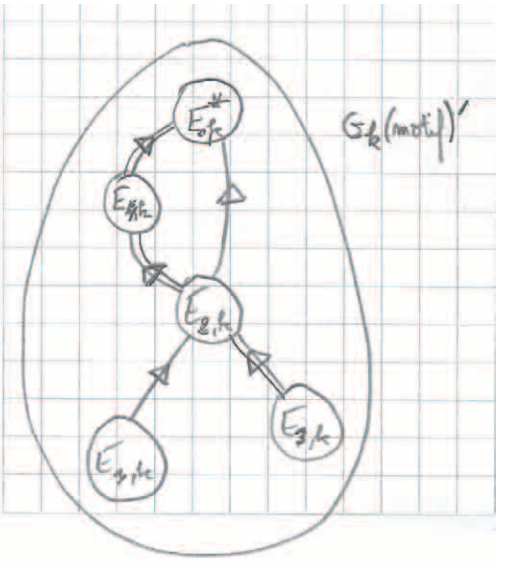


Fig-V-A2-11

La partie droite de la figure *Fig-V-A2-12* fait apparaître la représentation quantique d'un motif avec tous les chemins possibles et donc conservés. Sur chaque lien la lettre p_{ij} représente la probabilité partielle (en allant du nœud$_i$ au nœud$_j$)[3] de passer par un chemin. En sommant toutes les probabilités des différents segments d'un chemin, on obtient la probabilité du chemin tout entier.

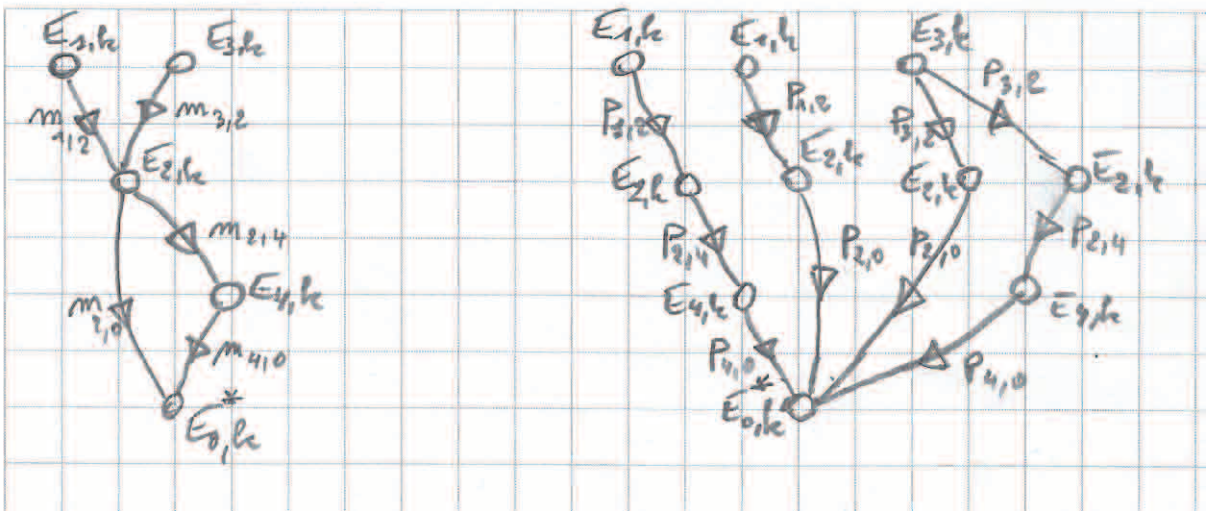


Fig-V-A2-12

3 Un état $E_{i,k}$ est une entité abstraite et purement topologique dans l'espace de représentation des connaissances. On va parler de nœud et non plus d'état.

Sur la partie gauche de cette même figure *Fig-V-A2-12*, on a la représentation, équivalente à celle de la partie droite, mais dans la base de connaissances et utilisant la gravitation informationnelle. Chaque lien entre deux états, se caractérise, en autres choses, par la lettre m_{ij} qui représente la masse relationnelle incidante de l'action du nœud$_i$ vers le nœud$_j$. Cette masse relationnelle intègre tous les chemins potentiels qui passent par ce lien (fragment de chemin). C'est une écriture condensée des chemins possibles sans avoir à les décrire tous.
Dans les deux représentations (quantique et relationnelle) la notion de conditions initiales (le positionnement et le nombre des serveurs dans l'exploration), n'est pas prise en compte volontairement, car inutile.

Construction de modèles

La représentation de connaissances nécessite la présentation de motifs en vue d'une première phase d'apprentissage qui fera apparaître de nouveaux modèles, suivie d'une phase de consolidation/évolution de chacun de ces modèles.

Construire le modèle 'A' revient à construire conjointement tous les modèles (les vingt-six lettres de l'alphabet) qui lui sont associés[4], afin que la différenciation puisse opérer efficacement. Cela revient à présenter des occurrences de tous ces modèles : c'est l'apprentissage.

Dans la mesure où un modèle peut évoluer constamment (la langue française, comme toutes les langues, en est un bon exemple), en réalité il n'y a pas de réelle opposition entre apprentissage et consolidation/évolution, mais plutôt un continuum. Un motif, qu'il soit nouveau ou parfaitement assimilé, sera exploré, puis son méta-graphe (MG(motif,k)') sera confronté aux méta-graphes équivalents dans la base. Le temps d'exploration d'une connaissance élémentaire (une lettre dans le cas du langage écrit) est très rapide, dans la mesure où elle ne fait pas appel à une géométrie complexe.
Si ce méta-graphe existe déjà, les liens le reliant à la base seront renforcés (augmentation de sa masse relationnelle), affaiblissant d'autant le rôle du temps sur un éventuel oubli de la connaissance associée dans la base.
S'il n'exite pas (première présentation) il sera positionné dans la base. Une absence de présentations ultérieures pourrait faire oublier, dans le temps, cette connaissance ; cela dépendra de la configuration de la *dérive des connaissances* (paramètres a_1 et a_2).
Le système de représentation de connaissances autorise la composition de modèles, par exemple pour former les mots du langage écrit. Un mot donné (par exemple *DERIVE*) devient un modèle d'un niveau supérieur à celui des lettres. C'est avant tout un modèle, donc il obéit au fonctionnement classique de l'espace de représentation. Le motif «mot»

4 Notre étude utilise l'alphabet, en simplifiant volontairement le contexte linguistique, afin de décrire certains mécanismes basiques de représentation et d'apprentissage, en jeu. Il faudrait ajouter dans l'apprentissage d'autres niveaux cognitifs du langage, comme nous les avons évoqués au chapitre II.

n'existe pas plus que le motif «lettre» dans cet espace, car il est représenté par un graphe relationnel entre les méta-graphes des différents motifs «lettre» et d'autres connaissances. En réalité l'état optimal ($E^*_{0,k}$), d'un motif donné, quelque soit son niveau de modélisation (lettre, mot…), est un nœud abstrait dans la base de connaissances. Ce sont les mécanismes sensoriels d'acquisition qui permettent la mémorisation «brute», hors représentation sémantique, de formes comme des traits pour la vision, de sonorités pour l'ouie, de gestes pour la gestuelle corporelle… Ces formes mémorisées sont à l'origine des mécanismes d'acquisition et de restitution qui stimulent l'espace de représentation pour une mémorisation sémantique et profonde.

UNE CHAÎNE CONTINUE D'ACQUISITION D'INFORMATION

À notre niveau, revenons sur les 'n' points associés aux motifs, qui sont candidats à être représentés dans la base de connaissances. Cela nous permet d'évoquer la chaîne continue d'acquisition d'information susceptible d'alimenter l'exploration, puis la représentation de connaissances.
Pour cela reprenons l'exemple de l'alphabet latin, comme nous l'évoquions au chapitre II, dans la partie II-D1.
Nous avions vu que les processus liés à la reconnaissance et à l'engrammage du langage naturel (parlé et écrit étant liés), posent le problème de la représentation des connaissances en machine.

Au départ il y a le signe graphique, les alphabets[5], puis plusieurs aspects se complètent, s'interpénètrent, s'articulent:

- la *morphologie* regroupe l'étude des formes attachées au sens, et plus particulièrement fait la distinction entre le concept de mot, trop vague, et celui de morphème, beaucoup plus opérationnel[6]; on trouve d'ailleurs des morphèmes lexicaux et des morphèmes grammaticaux[7];
- la *phonétique* étudie ce qui est émis et donc perceptible: les «phones»;
- la *phonologie* étudie comment la langue regroupe les phones en catégories appelées des phonèmes.

Dans ce contexte les morphèmes représentent l'unité minimale de sens que l'on ne peut diviser en unités plus petites. Au palier inférieur de la chaîne, on passe ensuite au niveau phonologique. Les phonèmes, basés sur les phones, sont totalement dépourvus de sens. Ils ont une fonction distinctive dans les morphèmes qui les

5 D'après Eric Havelock, dans Havelock, Eric A., 1981, «*Aux origines de la civilisation écrite en Occident*», traduit de l'anglais par E. Escobar Moreno, Paris, Maspero, cent-cinq pages, les trois conditions théoriques suivantes doivent être satisfaites simultanément pour pouvoir parler d'alphabet: «*il doit être rendu compte de tous les phonèmes de la langue, sans exception; le nombre de caractères ne doit pas dépasser un chiffre situé entre vingt et trente; les caractères ne doivent pas avoir un double ou triple emploi, leurs équivalences phoniques doivent être bien définies et invariables.*»

6 Dans «j'ai mal», si l'on descend 'mal' à 'ma' il y a perte de sens, donc 'mal' est un morphème. «redemanderons» comprend quatre morphèmes ('re'+'demand'+futur+1re personne du pluriel). «au fur et à mesure» ne comprend qu'un seul morphème, il s'agit d'une structure figée.

7 Les morphèmes lexicaux représentent une liste ouverte (adjectifs, noms…), alors que les morphèmes grammaticaux représentent une liste fermée (désinences des conjugaisons verbales, prépositions…) «Les boxeurs souffrent» est décomposé en sept morphèmes: dont deux lexicaux ('box' et 'souffr') et quatre grammaticaux ('le','s','eur' et 'ent').

intègrent[8]. Ils peuvent permettre le classement des sons, de la chaîne parlé selon une série de traits pertinents[9]. Les phonèmes, vus comme des unités distinctives minimales, produisent des différences négatives qui créent du sens. On parle de la double articulation du langage.

En partant du plus général (intégralité du discours d'un homme politique, comparaison de plusieurs œuvres d'un même écrivain…) on essaie d'aller vers l'unité minimale de sens. On peut analyser les différentes parties du discours, puis chaque partie en sous-parties, puis en phrases. On arrive à la première articulation : le morphème. La deuxième articulation concerne l'unité distinctive, le phonème, qui permet justement la différenciation entre plusieurs morphèmes.

En plus de l'encodage oral des morphèmes et/ou des mots, plusieurs langues ont ajouté un encodage graphique qui peut aller jusqu'à la production des alphabets[10]. Or il existe un lien entre les signes et le sens, représenté par les morphèmes, qui passe justement par les phonèmes. Il s'agit des graphèmes. Les lettres, ou les groupes de lettres (bigrammes et trigrammes[11]) sont des réalisations graphiques et contextualisées des graphèmes. Selon cette acception, un graphème est alors vu comme une entité abstraite[12]. Cette entité abstraite s'instancie par des allographes, qui ne sont que des représentations écrites concrètes d'un graphème en fonction du contexte (les lettres dans le voisinage immédiat).

---

8 « PÂLE » ET « PILE » N'ONT PAS LE MÊME SENS ET POURTANT LE PHONÈME /I/ N'A PAS DE SENS, C'EST UNE UNITÉ DISTINCTIVE. LE MORPHÈME 'NOUS' PEUT ÊTRE DÉCOMPOSÉ EN DEUX PHONÈMES [N U], EN LES CHANGEANT LE MORPHÈME CHANGE DE SENS.

9 « BARAQUE » POSSÈDE LE PHONÈME /R/, QU'IL SOIT PRONONCÉ AVEC UN [R] UVULAIRE OU UN [R] ROULÉ CE SONT DEUX RÉALISATIONS DU MÊME PHONÈME /R/.

10 GEORGES MOUNIN, DANS « L'ÉCRITURE SCRIPT, INTRODUCTION À LA SÉMIOLOGIE », AUX ÉDITIONS DE MINUIT, 1970, P. 137-144, A DÉCOMPOSÉ EN FORMES GRAPHIQUES ÉLÉMENTAIRES LES LETTRES DE L'ÉCRITURE LATINE POUR LES MAJUSCULES ÉCRITES EN SCRIPT. L'ALPHABET LATIN MET EN ŒUVRE UN SYSTÈME DE DOUZE TRAITS GRAPHIQUES MINIMAUX, DONT HUIT SONT DROITS, ET QUATRE ARRONDIS. IL CONSTATE ÉGALEMENT QUE SUR VINGT-TROIS LETTRES, DIX SONT FORMÉES DE TROIS TRAITS GRAPHIQUES, HUIT DE DEUX TRAITS, TROIS D'UN SEUL TRAIT, ET DEUX DE QUATRE TRAITS. 'A' = '/' + '-' + '\' ; 'B' = '|' + 'Ↄ' + 'Ↄ'… LA HASTE VERTICALE '|' EST UTILISÉE TREIZE FOIS ('B','D','E','F','H','I','K','L','M','N','P','R','T'), LE TRAIT HORIZONTAL '-' SEPT FOIS ('A','E','F','H','L','T','Z'),… « *TOUS CES TRAITS GRAPHIQUES MINIMAUX PERMETTENT DE FAIRE UN ENSEMBLE D'OPPOSITIONS DISTINCTIVES, QUI, DANS LEUR GRANDE MAJORITÉ, SONT DES OPPOSITIONS QUE L'ON QUALIFIERA DE PRIVATIVES, PARCE QUE L'UN DE CES MEMBRES SE CARACTÉRISE PAR L'ABSENCE DE L'ÉLÉMENT GRAPHIQUE QUI EST PROPRE À L'AUTRE. LA PLUPART DES LETTRES À DEUX TRAITS GRAPHIQUES S'OPPOSENT EN EFFET À DES LETTRES À UN SEUL TRAIT PAR LA PRÉSENCE D'UN TRAIT SUPPLÉMENTAIRE…* », DANS « *LES LETTRES DU LATIN : DESCRIPTION SÉMIOLOGIQUE, FONCTIONNELLE ET GRAPHÉMATIQUE* », P. 5, CHRISTIAN TOURATIER, UNIVERSITÉ DE PROVENCE. HTTP://WWW.PARIS-SORBONNE.FR/IMG/PDF/SEMIOLOGIE_FONCTIONNEMENT__GRAPHEMATIQUE-2.PDF.

11 LES GROUPES DE LETTRES SONT ASSOCIÉS À UN PHONÈME UNIQUE. ON TROUVE PAR EXEMPLE 'SS' POUR LES BIGRAMMES OU 'ŒU', 'OIN', 'AIN' OU ENCORE 'EIN' POUR LES TRIGRAMMES.

12 « *…ON PARLE DE GRAPHÈME, EN ENTENDANT PAR LÀ UNE UNITÉ GRAPHIQUE MINIMALE : « LE GRAPHÈME PEUT ÊTRE DÉFINI COMME LA PLUS PETITE UNITÉ (LETTRE OU GROUPE DE LETTRES) DE LA CHAÎNE ÉCRITE AYANT UNE RÉFÉRENCE PHONIQUE ET/OU SÉMIQUE DANS LA LANGUE PARLÉE » DANS CATACH, 1980, « L'ORTHOGRAPHE FRANÇAISE », PARIS, ÉDITIONS NATHAN, P. 334… POUR TRAVAILLER EN GRAPHÉMATIQUE COMME ON LE FAIT EN PHONOLOGIE, IL FAUDRAIT DÉFINIR LE GRAPHÈME COMME UNE UNITÉ GRAPHIQUE FONCTIONNELLE, QUE L'ON NOTERAIT ENTRE DEUX CHEVRONS, ET CONSIDÉRER QUE LES LETTRES NE SONT QUE DES ALLOGRAPHES OU DES RÉALISATIONS GRAPHIQUES DE GRAPHÈMES, TOUT COMME LES SONS NE SONT QUE DES ALLOPHONES OU DES RÉALISATIONS PHONIQUES DES PHONÈMES. SI DONC LA LETTRE 'S' CORRESPOND À UN PHONÈME /S/, QUAND RIEN DANS LE CONTEXTE NE LA CONDITIONNE, ON A AFFAIRE À UN GRAPHÈME QUE L'ON NOTERA <S>, CELUI-CI FAISANT CORRESPONDRE LA LETTRE 'S' AU PHONÈME /S/. MAIS QUAND, DANS UN CONTEXTE PARTICULIER PRÉCIS, CETTE MÊME LETTRE CORRESPOND À UN PHONÈME /Z/, ELLE REPRÉSENTE UN AUTRE GRAPHÈME, QUE L'ON NOTERA <Z>. ELLE N'EST PAS UNE VARIANTE DU GRAPHÈME <S>… MAIS UNE VARIANTE DU GRAPHÈME <Z>. ET INVERSEMENT, ON DIRA QUE LES DEUX LETTRES 'CE' SONT DES ALLOGRAPHES DU GRAPHÈME <S>, PUISQU'ELLES CORRESPONDENT AU PHONÈME /S/. ON ARRIVERA AINSI À LA CONCLUSION QU'EN FRANÇAIS, LE GRAPHÈME <S> EST REPRÉSENTÉ PAR PLUSIEURS ALLOGRAPHES, À SAVOIR LA LETTRE 'S', LE DIGRAMME 'SS' ENTRE VOYELLES, LA LETTRE 'C' DEVANT UN 'I' OU UN 'E', OU LE DIGRAMME 'CE'… SUIVANT LES CONTEXTES : LE GRAPHÈME DEVENANT AINSI VÉRITABLEMENT UNE UNITÉ FONCTIONNELLE COMPARABLE AU PHONÈME, ET LA LETTRE, UNE UNITÉ SUBSTANTIELLE, COMPARABLE AU SON… LA LETTRE 'S' SEULEMENT EST L'ALLOGRAPHE DE BASE QUI DONNE SON NOM AU GRAPHÈME <S>, PARCE QU'IL N'EST ENTRAÎNÉ PAR AUCUN CONTEXTE, TANDIS QUE 'C', 'CE', 'SS' OU 'T' SONT DES ALLOGRAPHES DE CE MÊME GRAPHÈME <S> QUI SONT CONDITIONNÉS PAR LEUR CONTEXTE GRAPHIQUE, DEVANT 'E' OU 'I', POUR LA LETTRE 'C', ENTRE DEUX VOYELLES POUR LES DEUX LETTRES OU DIGRAMME 'SS', ET DEVANT 'ION' POUR LA LETTRE 'T'…* » DANS CHRISTIAN TOURATIER, UNIVERSITÉ DE PROVENCE. HTTP://WWW.PARIS-SORBONNE.FR/IMG/PDF/SEMIOLOGIE_FONCTIONNEMENT__GRAPHEMATIQUE-2.PDF.

Par exemple le graphème noté <S>, qui représente le phonème /S/, possède quatre instanciations contextualisées : allographe$_1$ (la lettre 's' seule, sans voisinage), allographe$_2$ (la lettre 't' suivie par exemple de 'ion'), allographe$_3$ (le bigramme 'ss' entre deux voyelles) et allographe$_4$ (la lettre 'c' suivie de 'e' ou de 'i'). Autrement dit un graphème est une singularité, obtenue par émergence constitutive des différents allographes (instances) qui le définissent dans un environnement graphique donné (positionnement des lettres dans les morphèmes).
L'engrammage d'un texte passe par un processus de reconnaissance des caractères et réciproquement.
Cette étape peut se heurter à des ambiguïtés si l'on travaille sur l'écriture manuscrite. Cela peut être le cas pour le 'a' et le 'd'. Si l'on se base sur la décomposition de Georges Mounin (cf. la Note n°10) on voit que les lettres 'Y', 'V' et 'X' utilisent les mêmes traits graphiques ('\' et '/'), mais avec des positionnements spatiaux et des longueurs légèrement différents. Dans le cas d'une écriture manuscrite ces positionnements peuvent jouer un rôle très important dans la reconnaissance.
En fait pour décrire une lettre on s'aperçoit que l'on peut utiliser des règles de type oppositions distinctives, mais cela ne suffit pas. On pourrait recourir à des règles de positionnement dans l'espace des traits relativement les uns aux autres. Le problème étant de reconnaître une lettre parmi les autres «proches»; comme 'V' pourrait être considérée comme très proche de 'X', si l'on se réfère aux modifications spatiales à faire pour passer d'une lettre à l'autre.
Nous introduisons chaque lettre comme une forme canonique de l'alphabet. Pour savoir si une occurrence (variante écrite) d'un 'V' correspond bien au 'V' canonique et non au 'X' canonique, cela revient à appliquer à l'occurrence présentée les règles qui définissent la forme canonique 'V'. Dans notre cas précis on pourrait énoncer les règles suivantes : si les traits '/' et '\' se coupent à leur extrémité inférieure, alors nous avons un 'V'; s'ils se coupent en leur centre, alors nous avons un 'X'.
Chaque lettre définit un attracteur dans l'environnement d'assemblage des traits. La forme du bassin autour de chaque attracteur caractérise la zone d'incertitude qui entoure la reconnaissance de chaque lettre. Les bassins peuvent être plus ou moins adjacents les uns aux autres ('X' et 'V').
Concrètement cela signifie que l'on peut passer d'une lettre à une autre par simple transformation continue (glissement du '\' le long du '/'). Mais l'existence d'un très grand nombre d'occurrences possibles, quasiment infinie, peut nous amener à ajouter d'autres règles pour mieux prendre en compte toutes ces situations. Par exemple on pourrait énoncer pour 'V' la règle suivante : l'extrémité du trait descendant doit correspondre à l'origine du trait montant.

La richesse de l'acquisition des signes des lettres de l'alphabet, dans le contexte du langage naturel, ne fait que complexifier les modèles d'acquisition. Notre approche, basée sur la *dérive des connaissances* (MVB^{++}) doit permettre de tenir compte de cette complexité.
Nous allons nous focaliser ici sur une acquisition des lettres qui pourrait être faite à partir du motif écrit/dessiné, faisant uniquement appel aux traits visuels présentés.

Le motif devra être saisi, puis analysé, par exemple sur la base de la décomposition en traits élémentaires proposée par Georges Mounin (cf. la note n°10). Cette décomposition en traits élémentaires peut être transformée en une suite de points, par exemple les extrémités des segments repérés. Nous avons nos 'n' points-motif permettant de lancer le processus d'exploration (MVB^{++}), qui va produire le méta-graphe d'états (MG(motif,k)) correspondant.

Trois situations (cf. la figure *Fig-V-A2-13*) peuvent se produire :

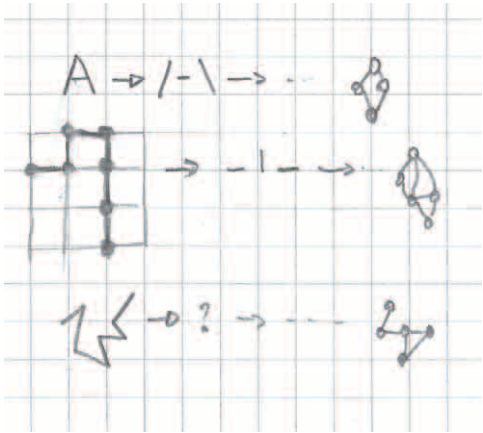


Fig-V-A2-13

- le motif correspond à un modèle déjà acquis dans la base, par exemple 'A'. Le modèle en question sera renforcé (augmentation de sa masse relationnelle et déformation de la zone de l'espace où il entretient des relations avec d'autres connaissances ;
- le motif ne correspond à aucun modèle de la base, mais il sera fréquemment présenté. Un nouveau modèle apparaîtra dans la base. C'est par exemple le cas de la signature graphique des téléphones/tablettes ;
- là encore, le motif ne correspond à aucun modèle de la base, mais il ne sera pas présenté à nouveau. Il sera donc vite oublié dans la base.

Nous partons de l'hypothèse que le motif, présenté à MVB^{++}, l'est sous une forme graphique, décomposable en 'n' points. Dans le cerveau humain (cf. la note n°12 dans la partie II-A2 au chapitre II) certaines parties de la mémoire joue ce rôle de mémorisation de formes (issues des différents sens), permettant de lancer des stimulations (rappel, engrammage…) vers d'autres mémoires plus profondes (mémoire sémantique…).
Dans notre approche de la représentation des connaissances nous pouvons aborder cette question comme pour le cerveau humain, avec un prétraitement qui transforme un motif en 'n' points, tout en gardant en mémoire l'intégralité du motif. Un lien peut être fait entre le motif, en tant que symbol ou entité homogène (étiquette), et sa représentation dans la base de connaissances.
Dans ce cas le nœud relationnel, correspondant à un modèle, pourra avoir comme donnée interne une «image» du motif.
Cette image du motif pourrait passer par un numéro qui serait associé au motif, par exemple, dans une table de hashage ou autre dispositif de correspondance.

Le chapitre VI, que nous n'allons pas écrire, avait donc pour objectifs :

- de réaliser MVB^{++}, en introduisant dans MVB la *dérive des connaissances* (une partie y a déjà été introduite) ;
- de travailler sur les mécanismes d'instanciation et de différenciation de modèles, en se basant sur les lettres latines majuscules. J'ai commencé à explorer, dans MVB, les méta-graphes des lettres 'A' et 'F'. Une étude de l'exploration des vingt-six lettres de l'alphabet aurait dû être entreprise pour mettre en évidence, grâce à la *dérive des connaissances*, les nombres d'états et de liens «utiles», ainsi que les profondeurs d'exploration 'k', nécessaires à la distanciation. Une meilleure

compréhension de la dynamique de représentation des lettres dans ce contexte de distanciation de modèles devrait nous donner des informations sur certaines caractéristiques de l'alphabet, comme, par exemple celle que rapporte David Havelock (cf. la note n°5) : «le nombre de caractères d'un alphabet ne doit pas dépasser un chiffre situé entre vingt et trente»;

- de comprendre les enjeux de la composition de modèles, en se rappelant que l'émergence doit être au cœur de cette approche, tout comme elle l'est dans la constitution d'une nuée d'oiseaux.

Conclusion

En terme de conclusion provisoire nous pouvons souligner le rôle présumé de la *dérive des connaissances* dans l'introduction d'une dimension supplémentaire (conduisant à la gravitation relationnelle) dans l'espace de représentation des connaissances de MVB, à l'image d'AdS pour TQC, dans la dualité holographique de Maldacena.
Je me suis amusé à parler de AdS-K et TQC-K pour faire le pendant quantique du modèle AdS-TQC dans l'espace relationnel de représentation des connaissances. Ce n'est pas une coquetterie, ni un effet de style, mais le signe, pour moi, d'une piste très sérieuse à explorer.

p.150 : «…Ainsi dans l'interprétation de Feynman de l'expérience des fentes de Young, les électrons individuels suivent chaque chemin possible allant du canon à l'écran. Un chemin emmène l'électron à travers la fente gauche, l'autre à travers la fente droite, une autre trajectoire peut le faire passer par la fente droite puis revenir par la gauche après un virage à cent-quatre-vingt degrés, puis à nouveau à travers la gauche. Chaque chemin possible - c.-à-d. chaque histoire possible - de l'électron, si absurde soit-il, doit être pris en compte, selon Feynman, et tous ces chemins contribuent à la figure que l'on observe sur l'écran. Le raisonnement de Feynman nous rappelle les routes alternatives que suggère un routeur GPS, au détail extrêmement inhabituel près - et profondément quantique - que, contrairement aux courses de taxis, les électrons prennent toutes les routes, intégrant par là même l'incertitude quantique. Comme l'a expliqué Feynman : "L'électron fait tout ce qu'il veut. Il va dans n'importe quelle direction à n'importe quelle vitesse, remonte le temps, à sa guise, et à la fin vous additionnez toutes les amplitudes [des chemins], ce qui vous donne la fonction d'onde."»

p.151 : «…Pour prédire la probabilité qu'un électron atteigne un point donné de l'écran, Feynman a associé un nombre complexe à chaque chemin, ce nombre déterminant non seulement la contribution à la probabilité totale, mais aussi la façon dont il interfère avec les chemins voisins. Par cette valeur, chaque chemin particulier se voyait attribuer les propriétés mathématiques d'un fragment de l'onde. Feynman a ensuite formulé une superbe équation, alternative à celle de Schrödinger, qui construit la fonction d'onde d'une particule en additionnant tous les chemins qui aboutissent à chaque point. Le schéma d'interférences sur l'écran provient alors

de l'enchevêtrement des trajectoires issues des deux fentes, que Feynman additionne. Mathématiquement, cela est dû au fait que le nombre complexe assigné à chaque chemin signifie que différents chemins peuvent se renforcer ou s'atténuer mutuellement, tout comme le font des fragments d'ondes qui se superposent…»

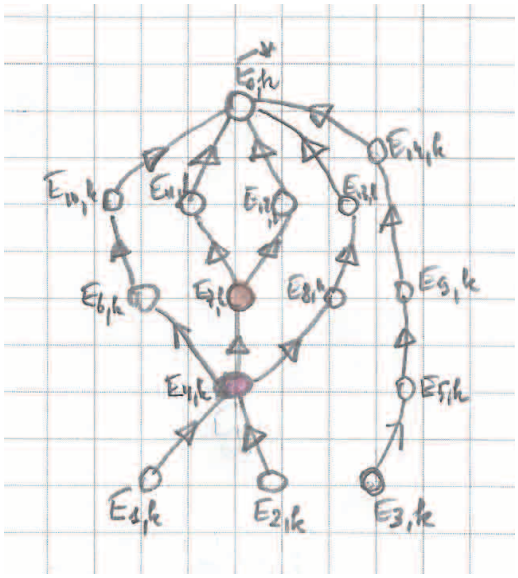


Fig-V-A2-14

Bien que je n'ai pas écrit la fonction d'onde de Schrödinger, ni même sa variante proposée par Feynman, adaptée à la *dérive des connaissances*, je suis persuadé que le modèle quantique est pertinent pour décrire les phénomènes en jeu dans mon approche de la représentation des connaissances. Condidérer une connaissance par une fonction d'onde relationnelle, c.-à-d. par la simultanéité de l'ensemble des chemins qui y conduisent dans l'espace de représentation, en affectant une probabilité à chacun d'eux, me paraît très adapté à la *dérive des connaissances*.
Cette fonction d'onde indique que certains chemins seront plus probables que d'autres, mais aussi que certains d'entre eux pourront se renforcer ou s'atténuer.
Cela indique aussi que, dans l'espace de représentation des connaissances, une connaissance est un tout, c.-à-d. tous les chemins possibles pour y arriver, avec des probabilités associées à chaque chemin, qui peuvent évoluer en fonction des observations/interactions faites sur la base.

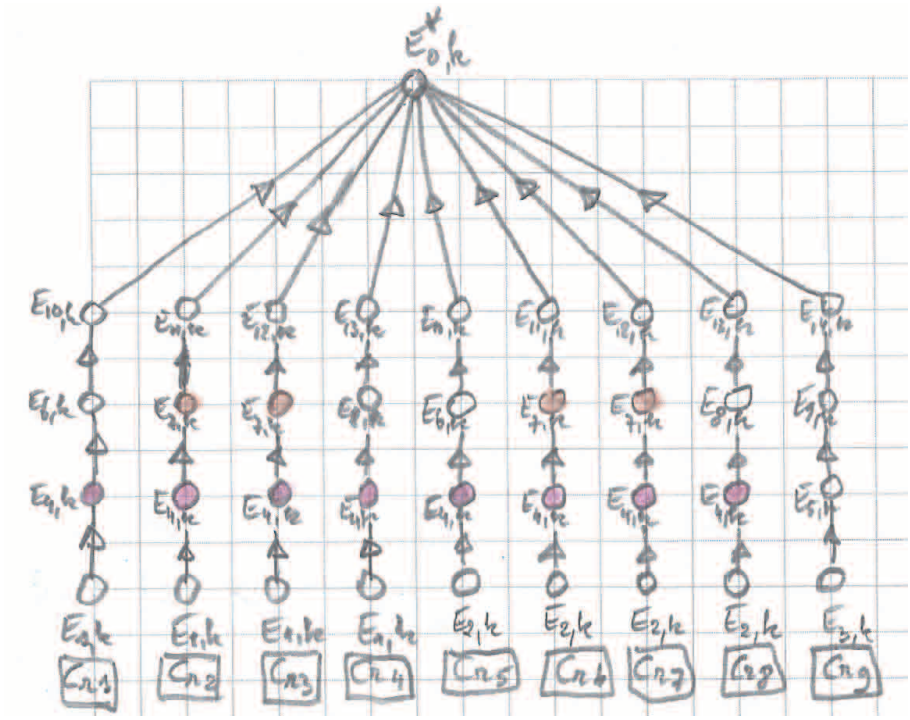


Fig-V-A2-15

On peut illustrer ceci avec l'exemple de représentation de connaissances présenté sur les figures *Fig-V-A2-14/15*.
Sur la figure *Fig-V-A2-14* nous avons la représentation d'une connaissance dans l'espace de représentation, avec les liens relationnels de la *dérive des connaissances*.
Cette connaissance est représentée par le graphe «utile» permettant de relier les points d'entrée (états) $E_{1,k}$, $E_{2,k}$ et $E_{3,k}$ à $E^*_{0,k}$. Pour alléger la figure, nous n'avons pas positionné toutes les masses relationnelles, attachées aux liens, et induisant la courbure de l'espace autour de cette connaissance. On observe, sur cet exemple, que les deux états $E_{4,k}$ et $E_{7,k}$ jouent un rôle particulier en rendant des fragments de chemins, communs à différents chemins.
La figure *Fig-V-A2-15* déploie la vision quantique de cette connaissance dans la base relationnelle. Neuf chemins différents et simultanés (Cr_1 à Cr_9) apparaissent.

Cet exemple montre plusieurs situations où les ondes, de chemins différents, se renforcent mutuellement, alors que d'autres s'atténuent les unes les autres.

Le nœud $E_{4,k}$ (en rose sur la figure *Fig-V-A2-14*) est commun aux huit premières trajectoires. Cela signifie que le passage par l'une de ces huit trajectoires renforce les sept autres en terme relationnel et donc probabiliste. Le nœud $E_{7,k}$ (en orange sur la figure *Fig-V-A2-15*) rend le lien $E_{4,k} \rightarrow E_{7,k}$ commun aux trajectoires Cr_2, Cr_3, Cr_6 et Cr_7, entraînant un renforcement de ces dernières lors du franchissement de n'importe laquelle d'entre elles.

À l'inverse la neuvième trajectoire Cr9 n'a aucun nœud, ni donc lien, en commun avec les huit premières trajectoires. On peut dire que cette trajectoire atténue les autres et réciproquement. Sauf à considérer que la probabilité de l'onde associée à cette neuvième trajectoire est proche de 1, ce qui signifierait qu'elle est très probable alors que les autres anecdotiques, la probabilité d'atténuation des huit premières trajectoires sur la neuvième est plus importante. En effet l'usage/excitation de n'importe laquelle des huit premières trajectoires atténue la neuvième.

On peut comprendre le lien intime et pertinent entre une vision quantique et la *dérive des connaissances* associée à l'espace de représentation.
Une connaissance est un tout mouvant dont les ramifications et la courbure de l'espace gravitationnel et relationnel donnent son extraordinaire plasticité. Je pense même que la notion de masse relationnelle, telle que je l'ai proposé dans la *dérive des connaissances*, n'est qu'une traduction gravitationnelle d'une approche probabiliste de cette question. Le modèle relationnel serait alors le pendant gravitationnel/classique du modèle probabiliste/quantique avec un changement d'échelle, lié à l'ajout d'une dimension AdS-K, permettant la composition de connaissances vers des niveaux de complexité supérieurs.

Pour terminer cette réflexion, on peut même faire l'exercice de pensée qui consiste à imaginer une expérience de fentes relationnelles, à l'image des fentes de Young pour les électrons, puis plus tard celle de Wheeler avec les photons et le store vénitien. Une même connaissance, comme celle des figures *Fig-V-A2-14/15*, passe à travers les deux fentes verticales de la première paroi. Sur la seconde paroi on peut observer le schéma d'interférence avec pour chaque bande verticale la probabilité que cette connaissance soit identifiée par cette bande. On peut traduire chaque bande du schéma d'interférence comme étant un chemin de représentation de la connaissance en question.

Si je pense à un modèle simple comme la lettre 'A' de l'alphabet latin, chaque fente du schéma d'interférence peut représenter chaque instance de cette même lettre 'A' permettant d'arriver au «modèle». Si je prends une connaissance sémantique plus élaborée, comme par exemple la catégorie «animaux», il y aura de nombreux chemins possibles pour arriver à cette connaissance.

On pourra mobiliser des chemins qui passent par «oiseaux», «souris»… Plus ces chemins seront nombreux et probables (plus leur «masse» relationnelle sera grande) et plus le schéma d'interférence sera riche et solide.

Sans reprendre tous les parallèles possibles, ou tout du moins ceux que j'ai exploré, je souhaiterais insister une dernière fois sur la dualité micro/macro, quantique/gravitationnelle des connaissances.

Dans la *dérive des connaissances*, l'espace de représentation peut effectivement être déformé (courbure) par les masses relationnelles. Toute connaissance peut être reliée par des chemins dont la dynamique obéit à l'accès exogène (à la base de connaissances) qui en est fait.
On peut faire un parallèle, à mon sens saisissant, entre l'accès à l'information dans la *dérive des connaissances* et le rôle de l'observateur en mécanique quantique. C'est en accédant que je fixe (renforce) l'histoire (chemin) qui va devenir l'unique chemin pour accéder au modèle, puis au modèle de modèle…

L'Estaca, le 11 octobre 2023.

VI INSTANCIATION ET DIFFÉRENCIATION DE MODÈLES

Usage de la *dérive des connaissances* dans MVB^{++}

Instanciation et différenciation de modèles

Composition de modèles

Ce chapitre se compose de trois parties dont une première qui consiste à introduire la *dérive des connaissances* dans le code de MVB, vu au chapitre IV. Pour la petite histoire, je n'ai jamais implanté ce modèle que j'ai entièrement spécifié lors de ma thèse soutenue en 1992. Il existe des débuts de codage du modèle dans MVB. Outre la quincaillerie relationnelle, il faudrait ajouter la gestion de la base (interaction avec le module d'exploration, un module de restitution…).
La deuxième partie consistera à engrammer toutes les vingt-six lettres de l'alphabet pour étudier le mécanisme de différenciation de modèles. En quoi un modèle se particularise de ses instances, au sein de la base de connaissances (méta-graphes, profondeurs…).
Cette deuxième partie permettra aussi de modeler l'espace gravitationnel de représentation des connaissances, en général, et en particulier celui des lettres de l'alphabet.
Enfin la troisième partie abordera la composition de modèles, en introduisant l'émergence de connaissances dans la base.
De nombreuses pistes ont été soulevées par ce mémoire. L'une d'entre elles concerne l'étude de l'espace des états dans C_nS_k, dont quelques pistes, données dans la partie III-D du chapitre III, pourraient compléter l'algorithme de contraction/expansion implanté dans MVB.

À [pour]suivre…

CONCLUSION GÉNÉRALE ET PERSPECTIVES VII

Il m'aura fallu quatre années pour mener à bien ce projet de Testament de recherche, dont les deux dernières sous le régime de la retraite, avec un statut de chercheur associé au GREYC.
En réalité, mon projet n'est pas complètement abouti puisque le chapitre VI, concernant la représentation des connaissances, n'est pas écrit, malgré des spécifications assez avancées proposées au chapitre V. Je pourrais même dire que son écriture pourrait faire le tome II du présent document…
Ce fut une expérience très intéressante qui se déroula de façon très intense, en particulier pour tous les programmes que j'ai écrit.
La boucle étant bouclée, je vais pouvoir passer à autre chose, lâcher prise en laissant s'éteindre définitivement le feu qui resta allumé, par moment en veilleuse, pendant plus de trente années.

Quelles perspectives donner à un testament ? Pour ma part je vais continuer à explorer les phénomènes émergents dans la peinture.
Concernant ce travail, je n'ai aucune idée de l'intérêt qu'il peut avoir pour des travaux à venir. Ce que je sais c'est que je l'ai fait avec une soif de comprendre, une honnêteté et une opiniâtreté à toute épreuve.
Les phénomènes émergents, dont les nuées d'oiseaux en sont l'un des exemples les plus parlants, restent et resteront pour moi d'un immense intérêt. Ils sont source d'imagination, d'émerveillement et titillent, comme rien d'autre, mon appétit de curiosité. Je leur en sais gré.

Une autre perspective à cette proposition pourrait tout simplement consister à faire des émules.

Personnellement j'ai pris un immmense plaisir à faire ce travail, dans des conditions professionnelles, puis de semi-retraite telles, que j'ai pu rassembler toute l'énergie nécessaire pour faire aboutir ce projet un peu fou.

En fin de carrière (dernières années) il peut être intéressant de faire un bilan sur un savoir-faire acquis au fil du temps, qu'il soit technique, technologique, manuel, intellectuel ou simplement professionnel. Suivant les cas cela pourrait prendre différentes formes, l'objectif étant de témoigner et transmettre aux générations qui viennent l'expérience d'une pratique d'une vie professionnelle.

Dans le cas d'un travail de chercheur, cela pourrait prendre la forme du testament que je propose, mais pas uniquement, bien sûr. Chacun étant à même de proposer le format qui lui conviendrait le mieux, y compris, dans certains cas, de ne rien proposer du tout.

Encre-temps

éloge de l'épaisseur du temps et de la lenteur

janvier 2022

«le papier ne refuse pas l'encre»,
proverbe populaire.

VIII

ENCRE-TEMPS, ÉLOGE DE L'ÉPAISSEUR DU TEMPS ET DE LA LENTEUR

Depuis une quinzaine d'années je travaille l'encre de Chine, dans une approche abstraite, sur toutes sortes de supports comme le papier (couché, offset, bouffant…) mais aussi le carton, la toile de verre intissée et même le verre.

Je travaille aussi avec des encres de couleur, du brou de noix, de l'encre (grasse) d'imprimerie et plus marginalement de la peinture à l'huile. Je réalise des formats très différents allant du A5 (14.85x21cm) à des encres pouvant faire cinquante mètres de long sur plus d'un mètre de large.

J'ai mené cette activité sous le pseudonyme de «Darius peintre-encrier» (dariuspeintreencrier.com) parallèlement à mes travaux de recherche au laboratoire GREYC de l'Université Caen en Normandie, sur l'émergence dans les systèmes complexes et artificiels.

Le temps se définit, dans mon activité de peintre-encrier, par rapport aux éléments que je convoque dans l'ultime objectif (ici «ultime» ne veut pas dire unique) de figer une représentation/trace a-temporelle, c.-à-d. interprétable émotionnellement dans le temps et dans l'espace.

J'ai réalisé un livre (non édité) appelé «encre-temps» dans lequel je n'ai pas théorisé mon

expérience avec le temps dans l'espace créatif, mais où je me suis attaché à montrer des exemples emblématiques de travaux illustrant cette expérience. On retrouvera de nombreux exemples tirés de ce livre dans les annexes (cf. les parties VIII-H) de ce chapitre.

Le propos de cet article, cristallisé dans le cadre de la conférence Rochebrune 2022 sur la thématique du «temps dans les systèmes complexes», est d'une part de développer une réflexion sur le temps, sous-jacente à mon activité plastique, hissant les pratiques artistiques au rang de systèmes complexes, et d'autre part de faire un lien plus général avec mes recherches scientifiques antérieures sur l'émergence dans les systèmes artificiels.

Après avoir posé les bases de mon contexte artistique de peintre-encrier (notions de trace, de médium, d'espace et de contraintes d'intervention…) je m'appuierai sur des réalisations concrètes (descriptions, photographies et vidéos) pour développer ce que j'entends par l'«encre-temps» et ses propriétés qui se distinguent de celles du temps «classique».

Je montrerai comment l'encre-temps donne de l'épaisseur au temps pour les interventions plastiques.

Dans ce contexte je parlerai :

- *de mouvement et de vitesse pour réaliser des traces, laissant apparaître, sous certaines conditions, des mondes autonomes dans lesquels des représentations d'éléments du futur, peuvent s'inscrire ;*
- *d'onde temporelle qui se propage et se base sur des granules temporelles allant du grain le plus fin, expérimentalement, à l'«infini»,*

où les propriétés classiques du temps, comme l'irréversibilité, la flèche…, peuvent être reconsidérées, ou au moins remises en question ;

- *de la Grande Vélaire (encre monumentale), dont sa réalisation hors du temps a nécessité une granularité temporelle relativement courte.*

Je montrerai à travers trois exemples (les deux premiers ayant une granularité temporelle de quelques secondes alors que pour le troisième elle sera «infinie») comment certaines propriétés temporelles sont impactées.

Je parlerai du repentir/pentimento en peinture (grande tradition), qui se joue de l'irréversibilité du temps sur une trace produite. J'aborderai deux voies que j'ai expérimentées parmi bien d'autres qui sont le recouvrement (partiel ou total) et le pliage.

Darius

Introduction

VIII-A

Le temps est une entité abstraite, un modèle, dans le sens où il nécessite une représentation, une singularité[1]. Il n'existe pas en tant que tel, mais par le biais des effets qu'il produit sur les choses et sur les personnes. Cette singularité se définit circulairement par les instanciations qui la définissent, en l'occurrence qui ont été observées, que ce soit lors d'expériences individuelles ou collectives.

À l'image d'une lettre de l'alphabet qui n'existe pas de manière tangible, mais qui se définit uniquement à travers les multiples manipulations que chacun opère (lecture/écriture/association…) avec elle, et comme toute singularité qui se respecte, le temps va lui aussi être co-construit par les expériences que chacun peut en faire. Il en découle des propriétés tout-à-fait intéressantes (irréversibilité, orientation, immatérialité, impossibilité de stockage, mesurabilité, épaisseur…).

Les activités sociales nécessitent une synchronisation entre les individus. Le temps, appelé temps objectif, devient une grandeur partageable, pourvu qu'on ait un référentiel[2] et une ou plusieurs unités de mesure.

---

1 Nous faisons référence ici à l'article intitulé « Singularité », présenté au chapitre III.

2 Dans le calendrier romain, les romains ont pris la fondation de Rome comme origine de la datation des années (an 753 avant notre ère). Actuellement nous utilisons pour les usages civils, le calendrier grégorien, qui reprend en grande partie la structure du calendrier julien de la Rome antique en vigueur jusqu'alors : les subdivisions en mois en semaines sont identiques, et le décompte des années se fait également à partir de l'*Anno Domini*, point de départ de l'ère chrétienne. Pour *Unix* le temps d'une machine se calcule à partir du nombre de secondes écoulées depuis le 1er janvier 1970 (époque *Unix*).

Le temps se mesure, se partage et se ressent à l'image de la température extérieure. Généralement plus long quand on s'ennuie, il se contracte quand on se passionne pour quelque chose. Ce ressentiment du temps convoque notre individualité, tout en faisant appel à des expériences partagées. On peut parler de temps subjectif.

À l'intérieur de notre propre individualité, plusieurs systèmes temporels distincts opèrent. On parle de temps biologiques multiples. À l'image d'un système informatique pour lequel le temps du CPU (entité calculatoire du processeur) est très différent du temps d'échange des données avec les périphériques, ce qui nécessite des synchronisations (via l'ordonnanceur), le système biologique humain doit lui aussi synchroniser les horloges internes qui le constituent.

Cette immatérialité qu'est le temps, joue donc un rôle crucial dans notre existence individuelle, mais aussi collective.

Michel Onfray, dans «Les formes du temps» donne une cartographie, loin d'être exhaustive, des temps visités : «*temps généalogique, temps immémorial, temps géologique, temps figé, temps spatial, temps primitif, temps anarchique, temps cyclique, temps repérable, temps séminal, temps végétal, temps tragique, temps circulaire, temps naturel, temps culturel, temps singulier, temps panthéiste, temps aléatoire, temps météorologique, temps climatologique, temps ontologique, temps négateur, temps destructeur, temps affirmateur, temps fédérateur, temps augustinien, temps transfiguré, temps ralenti, temps modifié, temps sculpté, temps entropique, temps agricole, temps chronométré, temps lent, temps pressé, temps irénique, temps magique, temps insipide, temps technologique, temps géorgique, temps féodal, temps alchimique, temps chimique, temps hédoniste, temps dionysiaque, temps spermatique, temps élémentaire, temps humain, temps ouvragé, temps magnifié, temps transcendé, temps sublimé, temps local, temps global, temps ponctuel, temps compressé, temps quintessencié, temps multiple.*» On pourrait ajouter le temps perdu et le temps retrouvé…

Dans le monde de l'art, la notion de trace et de temps a des propriétés et des exigences spécifiques.
Faire une trace implique un temps d'élaboration/réalisation de cette dernière, qui lui-même s'appuie, dans le cas particulier de l'encre, sur la notion d'*encre-temps* que nous verrons plus loin.
Une fois réalisée on passe dans un temps d'existence/conservation de la trace. Cette conservation n'interdit pas la transformation de la trace jusqu'à un point limite au-delà duquel la trace disparaît en tant que telle (palimpseste). On retrouve la même chose avec les nuées d'oiseaux dont leur existence n'interdit pas le va-et-vient d'oiseaux en son sein.
Conserver la trace implique, au-delà de la trace elle-même, une conservation du support qui recueille cette dernière. Qu'en est-il des traces au blanc de meudon sur une vitrine, lorsque cette dernière est brisée ?

Ou d'une peinture du 18e siècle dont les vernis l'auraient recouverte d'une épaisse couche brune foncée ?

Une trace artistique nécessite un temps social qui caractérise son apétance à être vue/perçue/ressentie/comprise, par un public dans le temps et dans l'espace.
Cette trace peut délibérément être contingentée dans le temps (installations éphémères comme les emballages de Christo ou les ombres de pluie d'Andy Goldsworthy, artiste britannique apparenté au Land Art, pour lesquelles il s'ést allongé sur le sol jusqu'à ce qu'il pleuve, laissant derrière lui la trace au sol de son corps). D'autres traces se voudront «éternelles» (sculptures en bronze).
Les œuvres sur papier, et plus spécialement photographiques, connaissent d'ailleurs aujourd'hui un engouement tout particulier pour la durabilité des encres et du support-papier dans le temps. On parle de label Digit Art avec des papiers 100% coton et des encres garanties cent ans en plein soleil…
Les arts numériques posent également le problème de leur durabilité dans le temps en conférant une importance maximale au support, l'encre numérique n'étant «rien d'autre» que le résultat de données numériques.
Dans notre cas nous restreignons notre propos à l'usage artistique de l'encre sur des supports divers comme le papier ou le verre.
Nous proposons, dans ce contexte, une autre approche du temps que nous appelons l'*encre-temps*, et dont l'existence prend sa source dans le travail artistique de l'encre. Pour cela nous nous appuyons sur de nombreuses expérimentations.

VIII-A1) Quelques définitions liées à la pratique artistique de l'encre

Le contact d'un médium, comme l'encre, sur un support construit un espace-temps avec ses caractéristiques propres (la dimension temporelle d'action, l'irréversibilité interactionnelle…).

La trace: c'est la production «dans» un certain temps, que nous appelons le «temps créatif», d'un espace, consécutivement au contact entre un médium et un support donné.

Il existe de très nombreux exemples de traces dans la vie courante. C'est le cas:

- du lavage des carreaux des vitrines de magasins (le récent travail photographique de Marie-Céline Navoux-Valognes sur des vitrines de magasins passées au blanc de meudon en était un exemple);
- de l'essuyage d'un tableau noir;
- des méandres d'un delta (cf. la photo ci-contre prise dans l'extrême Est Sibérien);
- ou encore du vol en nuée de canards, où des structures linéaires apparaissent furtivement dans le ciel.

L'encre de Chine (ou encre Sumi en japonais) est basée sur des pigments mélangés avec de l'eau. Suivant les supports utilisés, la concentration en eau, l'épaisseur déposée, les contraintes hygrométriques…), le temps de séchage varie énormément, entre quelques secondes, sur un papier absorbant, jusqu'à des durées plus importantes, sur du papier couché. Dans le cas de l'encre grasse d'imprimerie déposée sur du verre, ce temps de séchage peut être de plusieurs jours, voire davantage.

encre-temps

Le temps d'intervention pictural: ce temps correspond, dans la suite de cet article, au temps de séchage de l'encre sur son support. On l'appele aussi l'*encre-temps*. Il existe d'autres procédés d'intervention artistique, comme le pliage (cf. un peu plus loin la notion de «Pentimento»), qui peuvent ne pas tenir compte de l'*encre-temps*, dans les modalités présentées ici.

Ce temps d'intervention pictural ou *encre-temps* s'appuie sur le temps objectif, mesuré par exemple par un certain nombre de secondes. Il en subit certaines conséquences (orientation, irréversibilité…) jusqu'à un certain point que nous développerons par la suite. À l'image d'un oiseau volant dans une nuée qui, bien qu'obéissant à des règles spécifiques à la notion de vol en groupe, obéit également et avant tout aux règles basiques du vol d'oiseau seul.

Granularité de l'encre-temps: cette notion correspond à un intervalle de temps, c'est une mesure de l'*encre-temps*.

L'espace pictural: il s'agit d'un espace, ou monde, qui apparaît dans une trace après que celle-ci ait été faite conformément au principe de l'*encre-temps*.

Cette notion d'espace est purement subjective, un peu descriptive et surtout interprétative. Nous verrons sur quelques exemples, dont celui des lunettes, comment une interprétation des représentations/traces dans cet espace pictural peut se jouer du temps objectif.

La faille temporelle: cette notion reprend l'idée de granularité de l'*encre-temps*, donc de durée temporelle, pendant laquelle l'encre utilisée peut entrer dans un processus d'interaction avec son environnement et faire apparaître des espaces picturaux nouveaux.

épaisseur du temps

Au-delà d'un changement d'appellation, il s'agit ici d'introduire une nouvelle propriété fondamentale qui est l'*épaisseur du temps*, pendant laquelle le temps objectif alimente un temps de création qui se fige. Nous allons préciser cette notion et les propriétés qui en découlent.

L'onde temporelle de l'encre-temps: cette notion correspond au glissement spatio-temporel de l'*encre-temps* sur un support donné.

Ici on s'appuie sur l'idée qu'une trace est réalisée avec des outils chassant/déposant l'encre sur un support en invoquant le déplacement de ces outils.

VIII-A2) Mobilité dans la trace et apparition d'un espace primitif

De nombreuses techniques de peinture à l'encre existent, dont une en particulier, pratiquée au Japon, en Chine et dans de nombreux pays d'Asie, où l'eau, déposée préalablement sur le papier, diffuse l'encre dans les couches du papier.

C'est le cas du sumi sur le papier washi au Japon. Comme en témoignent les deux encres de cette page, que j'ai réalisées avec cette technique.

Dans notre approche, la trace est faite généralement sur une feuille de papier avec un outil qui chasse l'encre sur la surface de cette feuille.

Cela peut se faire en déposant l'encre sur le papier ou sur l'outil, en fonction des effets recherchés. On peut aussi mettre l'encre entre deux feuilles et utiliser un outil qui ne sera pas en contact direct avec elle…

L'énergie du geste, conditionnant les vitesses de déplacement de l'outil lors du traçage, joue un rôle déterminant dans le résultat obtenu.

Le sens du geste est libre dans l'espace, même si les traces résultantes sont sur une feuille en deux dimensions, la dimension spatiale, donnant un effet de relief, peut être très présente.

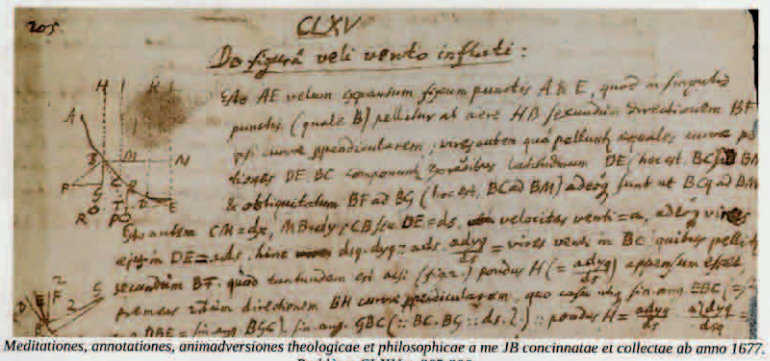
Cette liberté inclut la possibilité de revenir sur des zones déjà encrées. Le sens du temps est préservé dans la mesure où le geste se fait d'un instant 't' à un instant $t+dt$, et ainsi de suite.

Dans ces expériences la quantité d'encre déposée sur le support est suffisamment faible pour que son séchage de surface soit quasi instantané. Cela signifie que même en cas de recouvrement de la trace sur elle-même il n'y a pas d'interaction perceptible entre l'encre déjà déposée et la nouvelle.

La vélaire

Cette jolie forme que prend l'encre qui semble venir du ciel pour se poser au sol porte le nom de **vélaire**.

Voici quelques éléments de l'histoire de cette courbe.
Dans le numéro de mai 1690 des *Acta Eruditorum* (revue scientifique mensuelle éditée à Leipzig de 1682 à 1782), Jakob Bernoulli, mathématicien suisse, professeur à Bâle, propose un défi à ses collègues mathématiciens. Il s'agit de trouver quelle est la courbe suivie par un fil suspendu par ses extrémités et soumis à son poids.
Dans le numéro de juin 1691 des *Acta Eruditorum*, trois constructions différentes sont proposées conduisant à une même courbe appelée chaînette (*catenaria* en latin) : Gottfried Leibniz emploie la courbe des logarithmes, Christiaan Huygens se sert du calcul des sinus et Johann Bernoulli (jeune frère de Jakob) utilise la rectification de la parabole.
Pendant cette période, Jakob Bernoulli s'est posé des problèmes analogues, à savoir de déterminer des courbes que la nature nous met tous les jours devant les yeux et en particulier, celui de la forme d'une voile gonflée par le vent (*De figura veli vento inflati*).


205.
CLXV.
De figurâ veli vento inflati :

Meditationes, annotationes, animadversiones theologicae et philosophicae a me JB concinnatae et collectae ab anno 1677.
Problème CLXV p. 205-206

Jakob Bernoulli donne ce problème qu'il a résolu et ramené au problème de la chaînette, en défi à son frère Johann qui le résoudra en 1692 après que Jakob lui ait fourni des éléments intermédiaires.
Une certaine rivalité existait entre les deux frères. Animé par la volonté de clarifier la situation et d'attribuer à chacun ce qui lui revient, Jakob est amené à reconstituer à partir de la correspondance avec son frère, un récit circonstancié et détaillé de l'identification de la vélaire avec la chaînette. Ce texte fut publié dans le numéro de décembre 1695 des *Acta Eruditorum*.

652 DE VELARIA.

N. LXVI. V. Sed ut ad reliqua meletematum noſtrorum capita tranſeamus, atque aliquid etiam adjiciamus de *Curva Velaria*, quam eandem eſſe ſtatui cum catenaria, ſciendum, me nunquam mutaſſe ſententiam, ſed tantum diſtinxiſſe caſus. Dixeram, Menſe Maio

Jakob Bernoulli, *Opera Omnia*, 1744, p. 652-655, *De Velaria*

Didier Trotoux, juin 2019

VIII-B1) Présentation

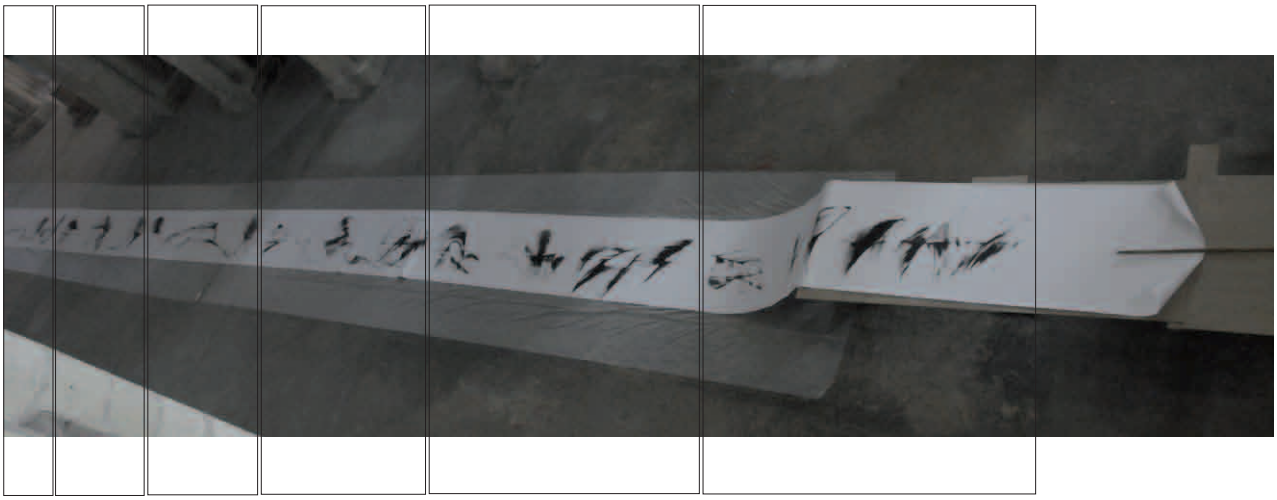
En dehors d'un changement d'échelle lié aux dimensions de la trace, nous sommes dans le cas vu précédemment où la mobilité dans l'exécution de la trace fait apparaître un espace primitif, en l'occurrence ici un alphabet, avec sa grammaire, dans un langage primitif corporel.

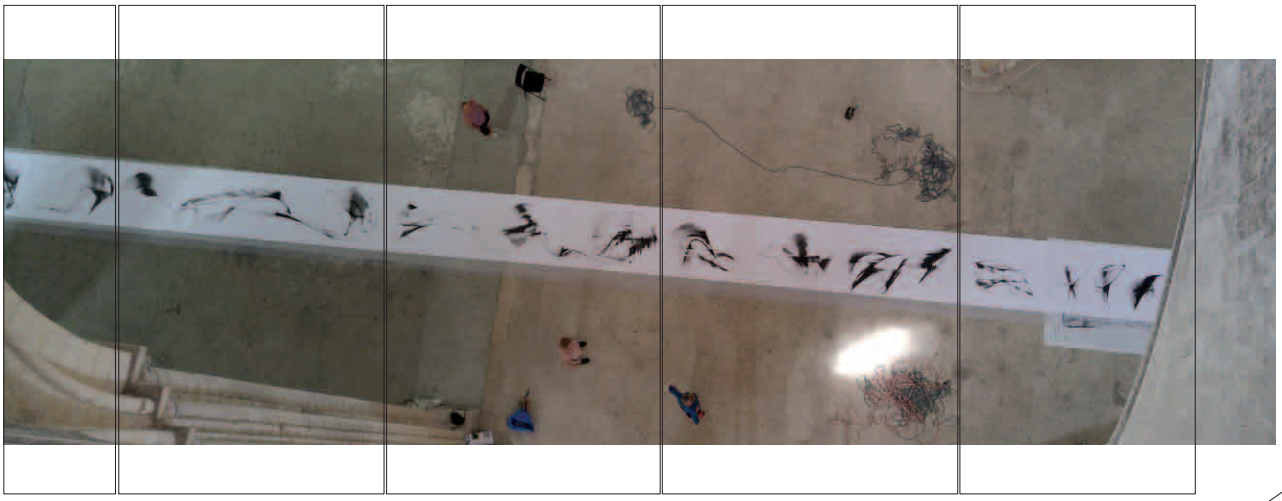
La Grande Vélaire est une encre monumentale de cinquante mètres de long sur cent-trente-trois centimètres de large, pour un poids total d'environ quatorze kilogrammes (en papier de deux-cent g/m^2). Elle a été réalisée *in situ* dans l'église Saint-Nicolas à Caen, où elle a été installée à dix-sept mètres dans le haut du chœur. Ses dimensions obligeaient une installation avec une partie conséquente posée au sol, lui conférant l'allure d'une vélaire.

VIII-B2) Particularités temporelles

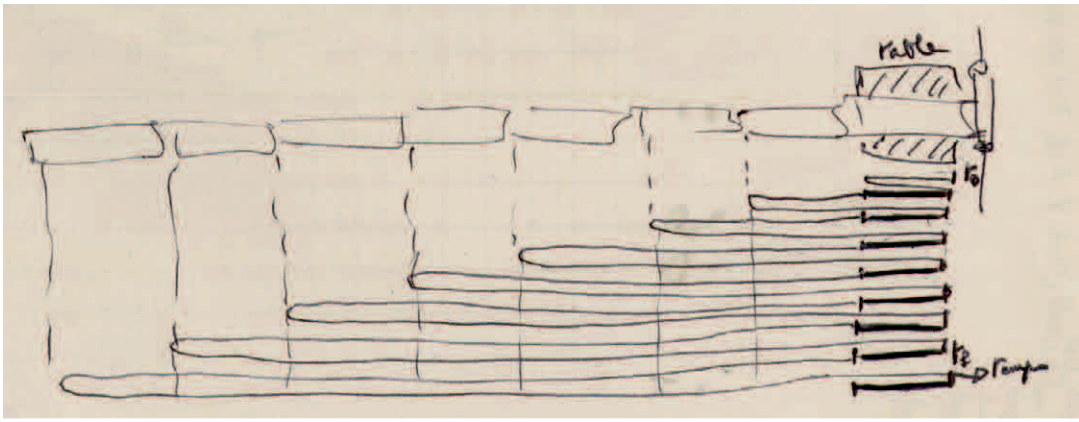
Cette encre est à l'échelle de l'édifice, mais hors norme pour un être humain. Sa réalisation, voulue comme un geste unique, sans répétition, nécessite un déplacement le long de la table de six mètres de long, sur laquelle le rouleau de papier fut progressivement déroulé. Comme le montre cette vidéo : «youtu.be/bHEpCHni5oU».
La trace adopte une forme de linéarité temporelle dans l'espace, marquant le papier d'une succession de huit caractères (chacun long de six mètres), tel un mot écrit à dessein en direction du bâtiment.
Outre le fait que les huit empreintes aient été faites au même endroit, sur la table, et que le papier, c.-à-d. l'espace, se soit déplacé à l'image de l'impression de caractères dans une machine à écrire, le point remarquable fut de considérer cette empreinte comme faite dans un seul geste. Le temps de réalisation s'est contracté au point d'être ressenti mentalement comme un instantanné, alors que, paradoxalement, en réalité il s'est dilaté par rapport à un geste équivalent sur une feuille de dimension habituelle, pour durer une douzaine de minutes.

Tout autre dispositif, en particulier dans mon atelier, aurait nécessité d'attendre le temps de séchage d'une séquence de six mètres avant de faire la suivante. Ce qui aurait rompu l'unicité spatio-temporelle du geste.





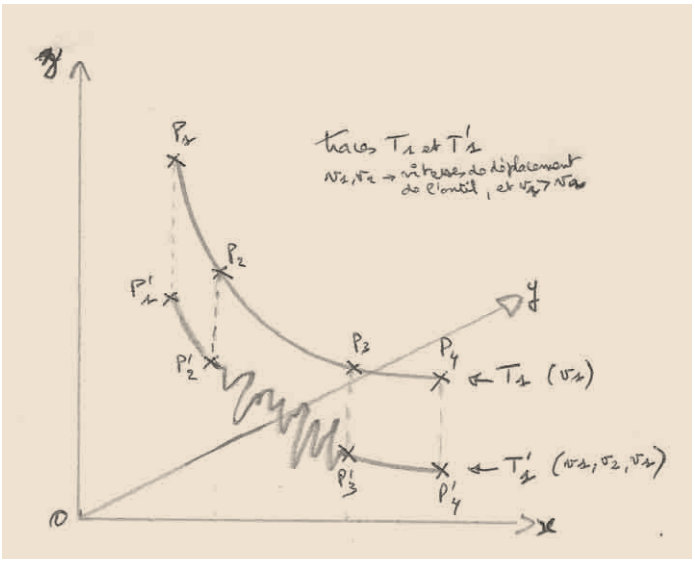
six caractères corporels



mes déplacements lors de la réalisation de la Grande Vélaire

Voici quelques chiffres liés à sa réalisation: durée de réalisation - environ douze minutes; distance totale parcourue - cinq-cent-quarante mètres, à la vitesse d'environ 2.7 km/h; vingt-six évènements temporels, dont huit (6x8=48 mètres) pour l'encrage, neuf (deux-cent-vingt-deux mètres) pour tirer le papier et neuf (deux-cent-soixante-dix mètres) pour revenir à la table d'encrage.

En ralentissant la vitesse d'exécution des mouvements lors du traçage, il apparaît des espaces picturaux d'une richesse inattendue. La lenteur crée de l'espace et télescope le temps dans des représentations anachroniques.



Le schéma ci-contre montre deux traces T_1 et T'_1 qui obéissent au même mouvement légèrement décalé dans l'espace, avec un ralentissement du geste entre P'_2 et P'_3, puis une reprise de la vitesse initiale entre P'_3 et P'_4. La courbe en zigzag illustre l'apparition d'un espace pictural spécifique.

En réalité au point P'_2 apparaît une bifurcation laissant naître un espace nouveau, imprévisible et d'une richesse qui peut souvent surprendre. L'intervalle de temps entre P'_2 et P'_3 s'apparente à une faille spatiale dans le déroulé temporel de la trace T'_1.

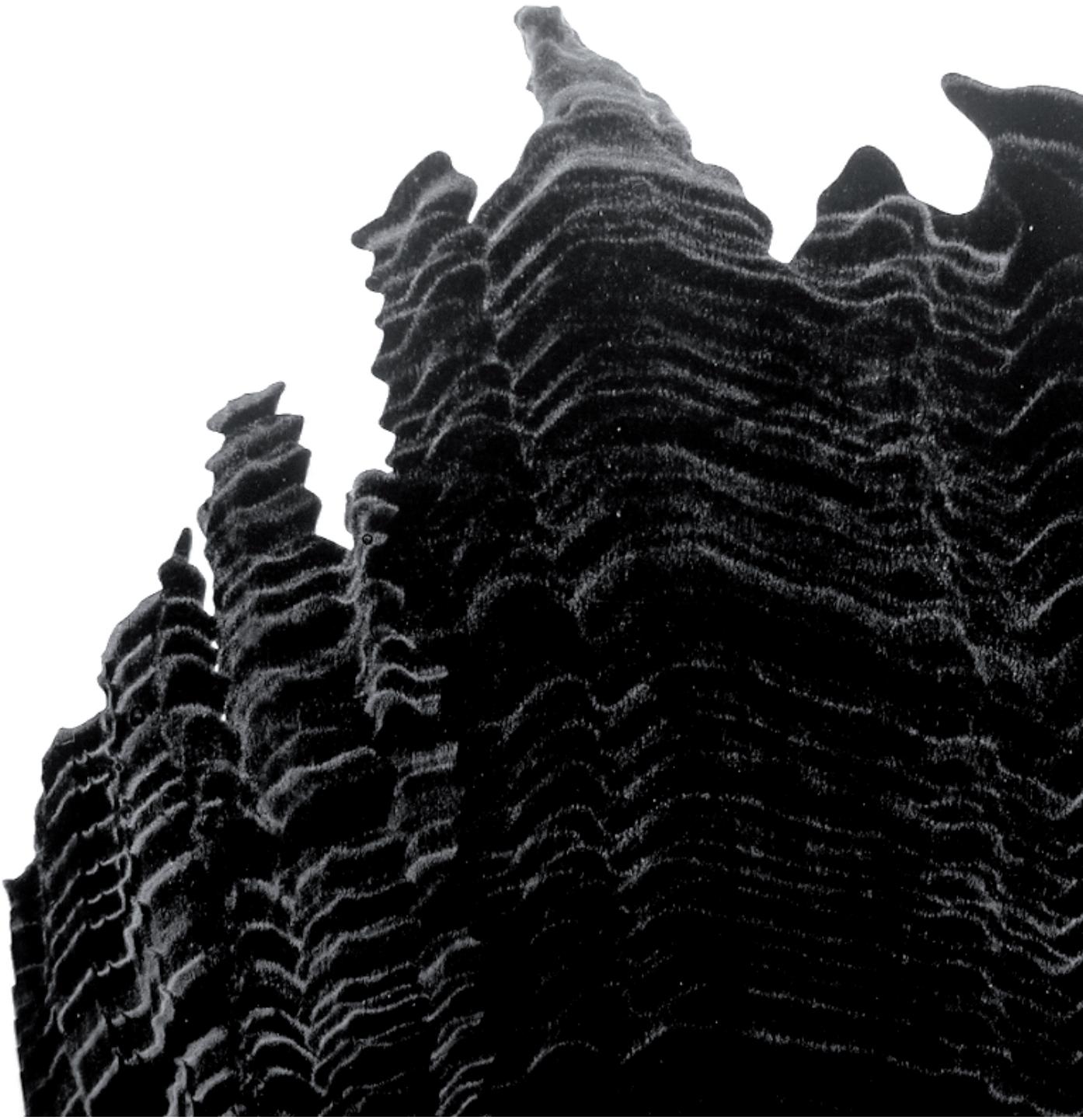
Cette lenteur ou ralentissement permet également de construire des formes très structurées, proches de la sculpture ou de l'architecture.
Un autre élément étonnant est que cette lenteur engendre/produit de la lumière dans la matière, mais aussi une représentation de la vie humaine, animale et/ou végétale.

Les exemples emblématiques sont légion comme le montrent les images en annexe (cf. les parties VIII-H).

VIII-C1) Apparition de matières, textures et formes architecturales

Les exemples proposés en annexe (dans la partie VIII-H3) montrent comment la lumière, la matière et la texture apparaissent. L'un d'entre eux fait penser à du granit éclairé par un soleil hivernal rasant, alors qu'un autre évoque un drappé généreux.

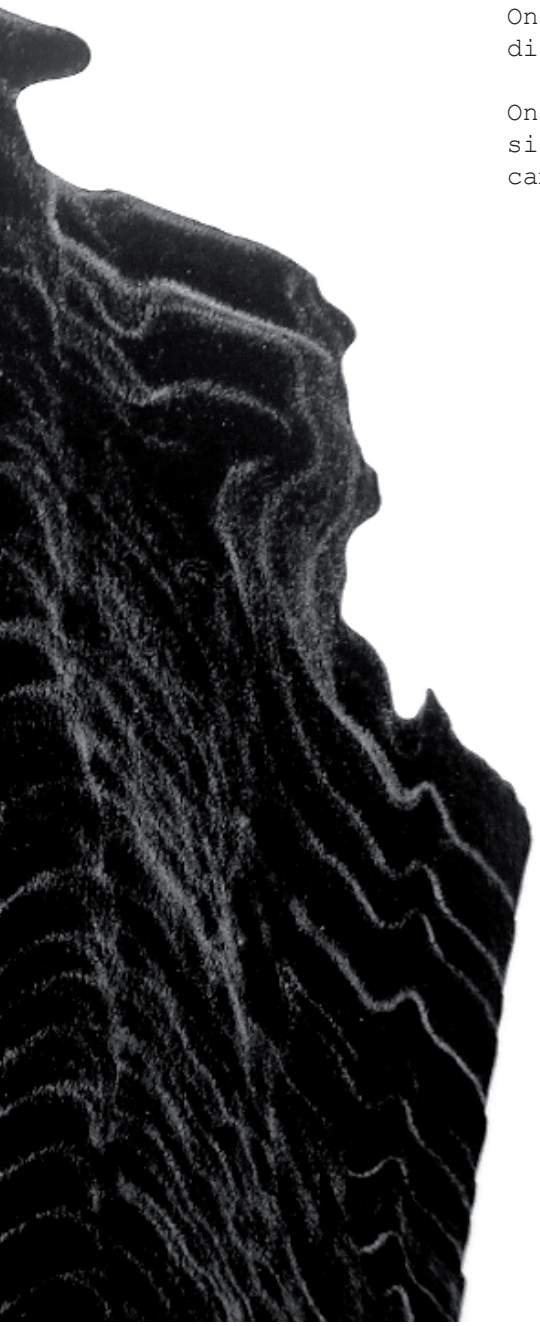
Ralentir le geste a mis en avant des formes et des structures architecturales d'une précision et d'une complexité remarquable; à l'image de l'encre ci-contre («à l'ombre de l'encre»), qui allie matière, texture et forme architecturale, mais aussi convoque la lumière, les transparences, le relief et la précision des lignes et des volumes.



VIII-C2) Apparition de paysages imaginaires

Cette lenteur du mouvement peut donc être à l'origine de bifurcations dont les effets sur la trace produite sont d'une richesse sans nom. On retrouve des formes classiques liées à l'écoulement, par exemple de l'eau de mer sur la grève à marée basse, ou à l'érosion des éléments sur des roches (trace ci-contre).
Ces paysages sont nombreux et variés, figuratifs ou abstraits, évocateurs et inspirants.

Dans les exemples présentés en Annexe (cf. la partie VIII-H4), on trouve des scènes très élaborées comme celle tout droit sortie des entrailles de la terre, ou encore celle où l'on voit une silhouette marcher dans ce que l'on pourrait prendre pour un champ de neige.



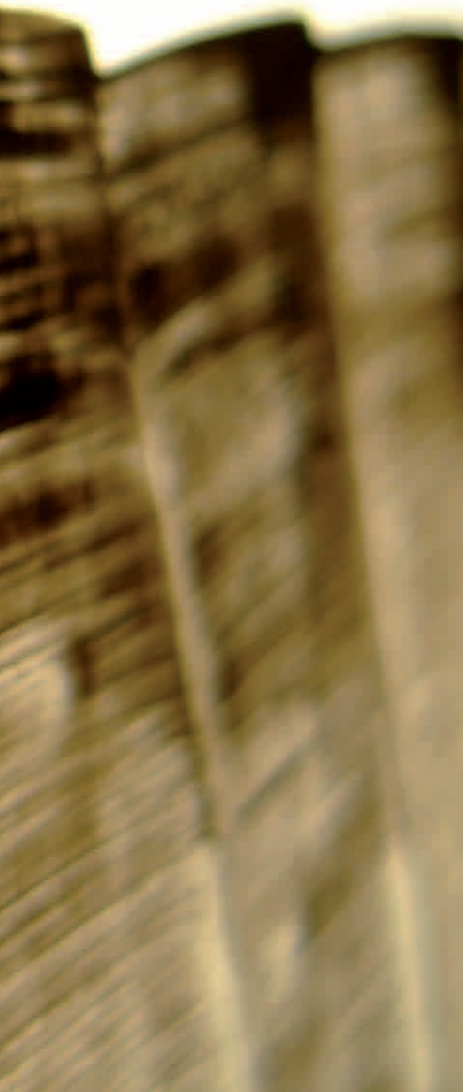
On peut aussi citer ce détail de vaisseau intergalactique digne des débuts de la science fiction (SF).

On peut également parler de la trace représentant une silhouette d'homme en veste marchant sur une route de campagne, où la perspective est très fidèlement rendue.

L'*encre-temps* est une durée temporelle pendant laquelle le dispositif de traçage autorise des interactions avec l'encre qui vient d'être déposée (au temps '*t*') avant qu'elle ne soit sèche (au moins en surface, au temps *t+dt*). Comme énoncé plus tôt, cette durée dépend de nombreux facteurs comme la nature du médium (par exemple l'encre), celle du support, les caractéristiques de l'outil, les conditions hygrométriques ambiantes…
Cette dépendance est d'ailleurs l'une des caractéristiques qui rend un(e) geste/trace unique, car difficilement reproductible dans les mêmes conditions.

Le mouvement associé au geste, qui donne la trace, est continu; son impact sur le support ne l'est pas nécessairement. Toutes les directions sont autorisées, autant que le support et l'outil le permettent.

Dans le cas de la Grande Vélaire, la trace est linéaire, toujours dans une même direction. Des coupures rythment chaque séquence de six mètres, imposées par les dimensions de la table d'encrage, mais aussi et surtout par la raclette, qui embarque une quantité d'encre suffisante pour couvrir cette surface sans variations inopportunes de tonalité. En réalité, la table a été construite en tenant compte des capacités de la raclette à encrer sans discontinuité.
Dans ce cas précis, l'encrage étant fait par une gestuelle très énergique et un déplacement corporel le long de la table, l'*encre-temps* n'était donc absolument pas convoqué.



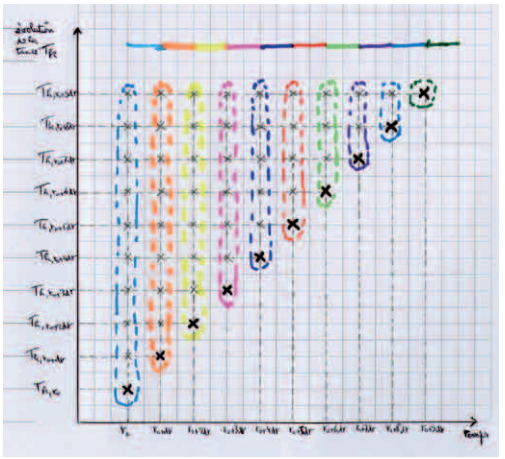
Regardons sur quelques exemples ce que permet un plein usage de l'*encre-temps*. En plus de la création d'univers incensés, dus en grande partie à la lenteur des mouvements - critère nécessaire mais pas suffisant -, l'exploitation de l'*encre-temps* apporte des éléments absolument remarquables. C'est comme si, au propre et au figuré, la lenteur posait un décor et l'*encre-temps* permettait d'y déposer des personnages/acteurs.

Avant cela, regardons de plus prêt ce qui définit cet *encre-temps*.

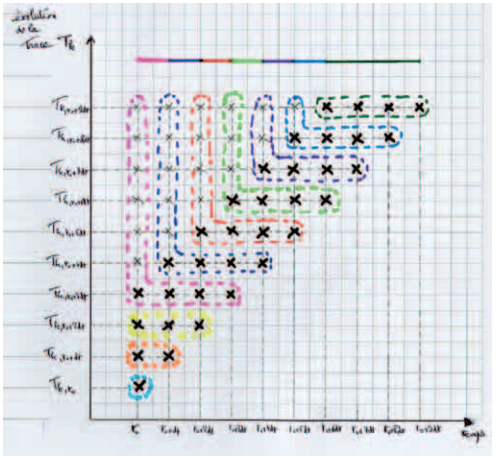
VIII-D1) *Encre-temps* et onde spatio-temporelle

L'*encre-temps* est une durée, un intervalle glissant, dont le front est le point de contact de l'outil avec le support lors de la réalisation de la trace. Cet intervalle est indépendant de la vitesse d'éxecution du mouvement associé à la trace. Il dépend de la quantité de média utilisé et des conditions dans lesquelles la trace est faite.
Si l'outil se déplace plus vite tout en déposant le médium dans les mêmes conditions que plus lentement, alors l'*encre-temps* sera constante.

Pour simplifier l'explication du phénomène, sans simplifier le phénomène lui-même, nous allons prendre le cas d'un déplacement uniforme (en vitesse, direction et conditions de dépôt du médium sur le support) de l'outil au contact du support.



*évolution de la trace T_k en fonction du temps (dix fois dt) sans recours à l'*encre-temps *(Δ=1).*



*évolution de la trace T_k en fonction du temps (dix fois dt) avec recours à l'*encre-temps *(Δ=4).*

Les éléments qui sont rapportés dans cette partie sont issus de la thèse de Muriel Cahen, intitulée «La structure du temps, ontologie et représentation», soutenue le 4 novembre 2015 pour obtenir le grade de docteur de l'EHESS en Philosophie.

P. 38

Newton-Smith (1980, III, 1, p. 48 et sq.) a établi une «topologie standard du temps» présentant un temps linéaire, c.-à-d. unidimensionnel, asymétrique et unique, et englobant tous les évènements…

Les propriétés formelles de la relation de précédence (entre les instants) impliquées par la topologie standard sont exposées par Newton-Smith sous la forme de sept axiomes. Les variables sont des instants. Ces axiomes sont l'irréflexibilité, l'asymétrie, la transitivité, la connectivité, la densité, l'infinité-1 : il n'y a pas de premier instant, l'infinité-2 : il n'y a pas de dernier instant.

Pour l'auteure, la topologie standard, basée sur des relations de précédence et donc de causalité, ne peut tenir compte de tous les phénomènes liés au temps, comme les évènements simultanés. Elle introduit les relations de recouvrement d'évènements et propose une combinaison avec les relations de précédence dans une double unité du temps. Pour cela, elle propose la notion de ligne temporelle. *Les relations de précédence au sein d'une même ligne temporelle seraient réductibles à des relations causales ou d'incompatibilité, alors que les relations de simultanéité ou de recouvrement seraient analysables en terme d'interaction réciproque ou de dépendance.*

Elle en déduit *que des évènements quelconques, appartenant à des séries temporelles distinctes, ne pourraient être établis qu'en considérant la combinaison de relations de recouvrement et de relations de précédence.*

Le recouvrement d'évènements, pour parler de la simultanéité d'évènements, fait écho à l'*encre-temps* où cette notion est bien présente.

P. 86-87

Dainton parle d'une conception naturelle du temps à partir des évènements passés, présents et à venir. Cette conception ne permet pas d'exprimer la simultanéité à la perception, ni de dater des évènements de façon précise (à une date donnée).

Schlesinger souligne que des sentiments de joie, de regret, d'espoir ou de crainte révèlent l'importance cognitive entre les trois «temps» (passé, présent, futur). Mellor ajoute *que nous projetons dans le futur parce que l'on peut l'affecter, alors que l'on sait que le passé est hors de notre portée.*

L'*encre-temps* correspond à la durée pendant laquelle l'encre qui a été déposée sur le support, au temps '*t*', n'est pas suffisamment sèche pour rendre inopérant toute interaction avec elle.
Elle est, en réalité, un intervalle de temps et donc d'espace, que l'on qualifie de glissant, à l'image d'une onde qui se déplace.

Formalisons le problème:

- $T_{k,t}$ est la trace T_k au temps '*t*', et donc $T_{k,t+dt}$ la même trace T_k mais au temps *t+dt*, avec *dt* l'unité de temps observable*;*
- dT_k est l'évolution de la trace T_k, entre '*t*' et *t+dt*. Elle se matérialise par une nouvelle surface couverte par le médium*;*
- ∀*t* dans la situation d'un déplacement uniforme (hypothèse explicative), nous obtenons: $T_{k,t+dt}=T_{k,t}+dT_k$.

Nous avons représenté l'évolution dans le temps de la trace T_k sans recours à l'*encre-temps* dans le diagramme de gauche de la page précédente, et avec recours à cette dernière dans le diagramme de droite.
Ces deux diagrammes montrent, par les croix en noir foncé, l'évolution de la trace à chaque pas temporel *dt*. Une croix en noir foncé représente donc dT_k, la nouvelle surface couverte par la trace T_k entre '*t*' et *t+dt*.

Le diagramme de gauche (ci-contre) montre clairement que l'outil ne revient jamais sur la trace déjà faite, ou s'il le fait, la partie de la trace impactée ne réagit pas au nouveau passage de l'outil, car l'encre (médium) est déjà sèche. L'interaction entre l'encre déjà déposée et le nouveau passage de l'outil (avec ou sans dépôt d'encre) est inopérante; il ne se passe rien d'autre qu'un recouvrement éventuel. Nous en reparlerons dans la section sur le repentir/pentimento en art.
Concrètement toute sous-trace de la trace reste inchangée dans la trace finale. Sur le diagramme de gauche, cela se matérialise par la présence, dans la trace finale T_k, de tous les segments colorés utilisés au cours du traçage.

En revanche, le diagramme de droite (ci-contre) fait apparaître l'*encre-temps*, que nous nommerons '*Δ*', durée pendant laquelle l'encre n'est pas encore superficiellement sèche. Cette durée permet à l'outil de revenir sur la sous-trace correspondante tout en continuant à développer la trace sur de nouvelles surfaces.

Dans cet exemple, nous considérons que *Δ=4dt*, où *dt* est l'unité de temps objectif, mesurable et significatif pour le tracé. Cela se matérialise par quatre croix consécutives (horizontal) en noir foncé. '*Δ*' représente une sorte de temps suspendu (*encre-temps*) pendant lequel *4xdt* de mouvement (déplacement de l'outil) autorisent une interaction entre la sous-trace passée correspondante et la nouvelle trace. L'interaction signifie une modification de la trace actuelle par un télescopage avec une trace pré-existante. Cette durée permet de mélanger passé et présent. Ce temps suspendu est une façon de donner de l'épaisseur au temps.

P.199
Les théories du présent spécieux, dont la plus célèbre est celle de James, introduit un lien direct avec l'*encre-temps*.
Le présent tel que nous en faisons l'expérience n'est pas un fil de rasoir mais plutôt une selle de cheval, avec une largeur propre, sur laquelle nous sommes perchés et depuis laquelle nous regardons dans deux directions dans le temps. L'unité de composition de notre perception du temps est une durée, avec une poupe et une proue, pour ainsi dire-une extrémité arrière et une extrémité avant. C'est seulement en tant qu'elles font partie de ce bloc de durée que la relation de succession de l'une à l'autre est perçue. Nous ne sentons pas d'abord une extrémité, puis l'autre, pour inférer ensuite de la perception de leur succession un intervalle de temps entre elles. Il semble que nous sentions plutôt l'intervalle de temps comme un tout dans lequel les deux extrémités sont intégrées.

P.221
En complément des théories du présent spécieux, certains auteurs, comme Mellor, défendent les théories de la mémoire dans lesquelles les évènements antérieurs sont remémorés et non directement perçus, induisant des relations de précédence entre les évènements. C'est ce qu'énonce Mellor de la façon suivante : *la distinction entre un contenu mémorisé et un contenu perçu reflète une précédence objective entre les évènements successivement représentés*, avec un bémol sur l'ordre des représentations en prenant l'exemple de la foudre et du tonnerre.
Nous touchons là l'expérience de la Grande Vélaire avec une réalisation courte dans le temps (environ douze minutes, déplacements compris) perçue comme un instantané (au sens présent spécieux-intervalle) et qui convoque des évènements possédant une certaine temporalité objective entre eux. La réalisation de l'encre, vécue comme un tout (geste) unique, est en réalité une suite d'évènements consécutifs (huit au total,sans compter les déplacements pour tirer le papier entre chaque encrage et ceux pour revenir à la table, vingt-six sinon), où chaque évènement présent procède de la mémorisation des évènements passés, constituant ainsi une expérience mémorielle de la construction d'un temps suspendu, dont la finalité conduit à une trace unique et non répétitive. La décomposition de la trace en huit morceaux chronologiques disparaît dans sa restitution finale/globale.

P.230
Faute de pouvoir fonder la représentation de la succession sur la succession des perceptions (c'est le cas de la foudre et du tonnerre), *des représentations ou sur la structure de ces dernières, on pourrait fonder les relations temporelles et leur représentation sur une autre relation, la relation causale. Cette idée est notamment développée par Kant, qui considère la causalité comme un principe* a priori *de l'entendement, qui organise l'expérience et permet ainsi de considérer que les évènements entretiennent objectivement des relations causales, même en l'absence de perception directe de telles relations. Bien plus, ce principe, stipulant que toute cause précède son effet, permet à un sujet de considérer que les évènements entretenant des relations causales entretiennent objectivement des relations de précédence. C'est notamment ce qui permet, selon lui, de distinguer entre la succession des perceptions et des représentations des parties d'une maison et celle des représentations des différentes positions d'un bateau: les positions du bateau, mais non les parties de la maison, entretiennent des relations causales et sont directement expérimentées en tant que telles. Par conséquent, les premières mais non les secondes sont directement représentées comme entretenant des relations objectives de précédence*.

La Grande Vélaire correspond bien au bateau et à l'expérience causale, en revanche ça n'est pas le cas pour l'*encre-temps*, où l'épaisseur du temps efface toute relation causale entre les évènements en interaction (dont les effets se fondent). L'*encre-temps* apparaît comme une faille spatio-temporelle, un trou noir des relations causales et donc du temps (suspendu)…

onde spatio-temporelle

En dehors des trois premières lignes, de la figure précédente (avec l'*encre-temps*), qui correspondent respectivement à T_{k,t_0}, T_{k,t_0+dt} et T_{k,t_0+2dt}, on observe sur ce diagramme le glissement de l'*encre-temps*, que nous appellons l'onde spatio-temporelle. Elle apparaît sous la forme d'un plateau de 'Δ'(ici 4) croix noir foncé, qui glisse avec le temps.

En considèrant, dans notre exemple, chaque croix comme un évènement, c.-à-d. une modification effective de la trace, nous avons $n(n+1)/2$ évènements, avec $n=10$ (le nombre de pas d'unité temporelle dt, soit de t_0 à t_0+9dt), donc cinquante-cinq évènements (cinquante-cinq croix sur la figure).

Lorsque que l'on ne considère pas l'*encre-temps*, ce qui correspond à $\Delta=1$(l'encre sèche instantanément), nous avons 'n' évènements en tout et pour tout (dix croix en noir foncé sur le premier diagramme).
À l'inverse, si le médium ne sèche presque jamais (équivalent à un temps de séchage très long), dans ce cas extrême, que nous verrons avec l'exemple de l'encre d'imprimerie sur une plaque de verre, le nombre maximum d'évènements est égal à $n(n+1)/2$ (ici cinquante-cinq, avec $n=10$).

Entre ces deux situations extrêmes nous avons le cas de la deuxième figure et l'onde spatio-temporelle. On obtient le nombre d'évènements (croix en noir foncé) par la formule suivante: $[\Delta(\Delta-1)/2]+[(n-(\Delta-1))\Delta]$ (ici trente-quatre avec $n=10$ et $\Delta=4$).

Pour résumer cette notion d'onde spatio-temporelle et d'*encre-temps*, il nous faut faire la distinction avec une approche linéaire.

Dans une approche linéaire, où le temps et l'espace de la trace obéissent à une relation de causalité/précédence, la trace T_k est une succession ordonnée de sous-traces. Dans ce cas, la trace est la somme des sous-traces à l'image du modèle du bateau de Kant (cf. les références ci-contre). On peut comparer cette approche linéaire à l'écriture d'un texte dont la version finale est faite d'une succession de phrases, parties et chapitres.

En revanche, dans le modèle de l'*encre-temps*, la trace complète n'est plus exactement la somme des sous-traces qui auraient été faites suivant une relation de causalité/ précédence. L'épaisseur du temps induite par ce modèle introduit des relations de précédence et de simultanéité complexes entre les évènements (sous-traces), rompant ainsi une relation de causalité entre eux. On se retrouve davantage dans le modèle de la maison de Kant, qui empêche toute déduction de règles de précédences et donc la construction d'une perception du temps, à la simple lecture de la trace et de ses parties (sous-traces).

Pour reprendre la métaphore de l'écriture d'un texte, ce modèle de l'*encre-temps* correspond au fait qu'en lisant un texte dans sa version finale, il est impossible d'en connaître les étapes intermédiaires (phrases, parties, chapîtres réécrits) qui ont disparu tout au long du processus d'écriture.
Cette métaphore littéraire nous permet de souligner l'idée que l'*encre-temps* semble exister dans beaucoup d'autres domaines. En effet le support reste métaphoriquement le papier et le médium l'encre. L'*encre-temps,* en littérature, devient le temps pendant lequel une phrase, une partie ou un chapitre peuvent être réécrits sans remettre en question les autres parties du texte.
J'avais écrit en 1990 (TOOLS'90) un article intitulé «An Experiment in Prototyping Using the Object Model as Structuring Agent» dans lequel j'évoquais un principe de structuration du code, fondé sur une intégration progressive du nouveau code écrit dans des strates «objet».
code-temps
Là encore, il s'agissait en réalité de formaliser le *code-temps*, durée pendant laquelle le nouveau code peut subir suffisamment de modifications pour ne pas être sanctifié dans le modèle objet sous-jacent (classes/méthodes/héritage…).

VIII-D2) Plafonds de cathédrale et autres situations

Dans cette partie, nous allons illustrer les concepts d'*encre-temps* et d'onde spatio-temporelle appliqués à la pratique de l'encre de Chine sur papier.
Le premier exemple emblématique de cette pratique est celui lié aux «plafonds de cathédrale». Chasser l'encre à la surface du papier nécessite un outil dont la principale caractéristique est d'être rigide. Il ne faut pas confondre ici chasser avec déposer.
À l'image des techniques asiatiques recourant à l'eau pour chasser l'encre dans la structure du papier, obtenant ainsi des variations infinies de nuances de gris, le travail de l'encre à la surface du papier nécessite, pour obtenir les mêmes variations, d'écraser ou de pincer l'encre sur le papier pour jouer sur l'épaisseur de la trace laissée et donc sur sa tonalité. Il faut préciser que plus l'encre de Chine déposée est peu épaisse et plus sa tonalité va vers le (gris) clair.

À cette propriété s'en ajoutent d'autres, importantes en peinture, dont celle qui rend brillante l'encre dès lors qu'une certaine épaisseur a été déposée sur le support.
Le peintre et sculpteur japonais Kodo Okuda (photo ci-contre) développe une technique originale et pointilliste pour obtenir des aplats d'encre à la fois noirs et mats, ce qui est antinomique.

L'outil, outre ses capacités à chasser l'encre, peut retenir le médium, ici l'encre. Il opère alors comme un stylo ou un pinceau.
C'est le cas de l'outil utilisé pour réaliser la Grande Vélaire, vue précédemment.
En plus, si cet outil peut être déformé lorsqu'il chasse l'encre (déplacement), on obtient des effets

ou des espaces picturaux très intéressants, à l'image des plafonds de cathédrale (grande encre de la double page précédente).
Dans la partie «Éloge de la lenteur», nous avions présenté une encre produisant l'effet du granit éclairé par un soleil hivernal. L'outil utilisé, pour cette réalisation, est un balai d'essuie-glace de voiture, qui est à la fois rigide et déformable.
Sur la trace ci-contre, se pose un autre problème vis-à-vis du temps. En regardant de près cette trace, on peut apercevoir au centre une femme au chignon avec un manteau noir et blanc dont le visage, tourné vers la gauche, porte des lunettes foncées. À ses pieds, on peut distinguer une bête de compagnie regardant dans la même direction que la femme aux lunettes.

Cette trace mobilise un outil simple (rigide), de l'encre et du papier. Elle aurait pu être faite bien avant l'invention des lunettes. Cette scène est entièrement subjective dans son interprétation et sa représentation.

Nous avons potentiellement un télescopage temporel entre la réalisation de la trace et ses interprétations possibles.
Cela questionne sur le fait que des traces peuvent porter en elles des représentations d'objets inexistants au moment de leur réalisation, mais qui prendraient sens dans un futur plus ou moins lointain. Rien n'interdit de penser que dans certaines traces figurent déjà des représentations d'objets du futur qui n'existent pas encore (ne sont pas encore interprétables). Simple coincidence ou réel télescopage temporel ?

Une grande partie des exemples présentés en annexe VIII-H5, exploitent l'*encre-temps* en utilisant tout simplement les doigts qui tiennent l'outil en mouvement, pendant le traçage. En laissant traîner mes doigts sur l'encre encre fraîche juste derrière la trace (sur la surface liée à l'onde spatio-temporelle de l'*encre-temps*), une interaction opère et donne naissance à des personnages, des situations, des objets, des animaux, de la flore… Le théâtre d'une représentation aussi étonnante que variée installe ses acteurs sur la scène et les photographie dans une trace qui devient intemporelle.

VIII-D3) Pressage et arrachage

Il existe de multiples techniques pour exploiter l'*encre-temps* qui ne dure que quelques secondes, avec l'encre de Chine sur du papier couché (dont la propriété est de ne pas être absorbant).
L'une d'elle (cf. la trace ci-contre), différente de celles utilisées dans les exemples précédents, consiste à superposer face contre face deux traces, fraîchement

faites sur deux feuilles de papier distinctes et à procéder à quelques manipulations (écrasement…) avant de séparer les deux feuilles.

Si cette opération se fait correctement, c.-à-d. pendant l'*encre-temps* (au-delà, les deux traces seraient collées définitivement l'une avec l'autre), alors on peut faire un arrachage superficiel d'une trace sur l'autre. Cette interaction permet d'obtenir des effets de blanc dans des zones noires de la trace, ou des effets de transparence/ voilage comme c'est le cas sur la trace de la page précédente.

six arrêts sur image du flux-trace infini
réalisé avec de l'encre grasse d'imprimerie sur du verre

VIII-D4) Un exemple d'*encre-temps* « infinie »

En choisissant les propriétés du médium et du support, il est possible de faire tendre l'*encre-temps* (‘Δ’) vers l'infini. L'épaisseur du temps de création n'a plus de limite. L'acte de tracer échappe totalement à toute relation causale (précédence temporelle) et, dans une certaine mesure, à toute forme d'irréversibilité puisque chaque trace produite est sans cesse transformée en la suivante.

Pour illustrer ce point, prenons l'exemple de l'encre grasse d'imprimerie sur un support en verre (cf. les photos ci-contre). L'encre d'imprimerie met un temps très long à se figer à cause de l'huile qui constitue son principal composant. Appliquée sur du verre, aucune absorbtion par le support n'est réalisée. Pour peu que l'épaisseur de l'encre soit suffisante, il faudrait attendre très très longtemps pour atteindre la fin de l'*encre-temps* (séchage de l'huile).

Nous avons réalisé une expérience qui a été filmée pendant une vingtaine de minutes sans le moindre séchage de l'encre, ni fin de l'*encre-temps*.

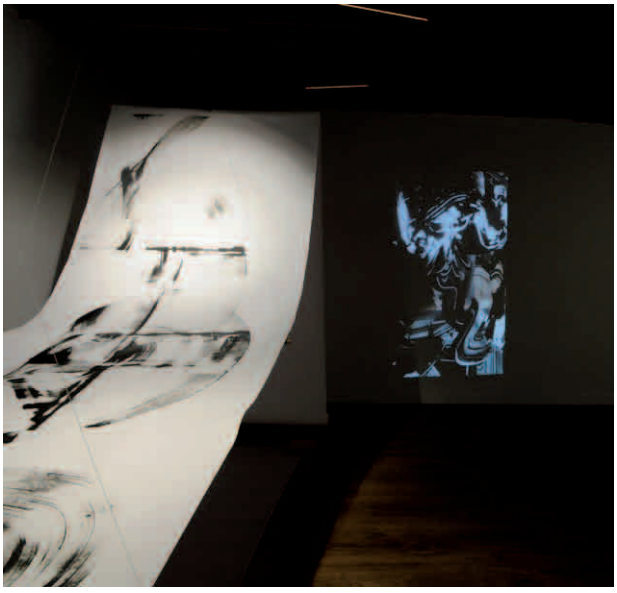
Cette expérience fut intéressante car elle rendait la notion de trace intangible. En effet, l'outil utilisé pour modeler sans fin l'encre étant une raclette, il apparaissait un flux de traces qui se déformaient continuement, les faisant apparaître et disparaître à peine entr'aperçues.

Il faut bien comprendre ici qu'il ne s'agissait pas de recouvrir une trace au temps ‘t’ par une autre trace au temps $t+dt$, mais de transformer la première trace en une seconde par un processus continu. C'est le propre de l'*encre-temps* qui fonctionne sur l'interaction du médium et des outils sur le support.

flux-trace

Nous étions confronté à un flux-trace exactement comme dans un film constitué d'images qui s'enchaînent.

L'*encre-temps* infini peut être vue comme une faille spatio-temporelle, un trou noir, où le temps de la création s'arrête dans une épaisseur infinie. Le paradoxe de cet arrêt sur le temps est de rendre la trace en continuelle évolution, à l'image des nuages dans le ciel. Elle ne s'arrête jamais de se transformer. Cet épaisseur du temps donne la possibilité de transformer une trace en un flux-trace générateur d'une infinité de traces.



Nous étions donc face à ce dilemme : à quoi sert de générer une infinité de traces qui n'en font qu'une, la dernière ?

Pour échapper à ce problème, nous avons utilisé le subterfuge de la vidéo qui nous a permis de faire, après coup, des arrêts sur image comme autant de traces générées pendant l'*encre-temps*. Ces traces deviennent virtuelles, puisque seule la dernière existe réellement, si tant est qu'on puisse la conserver malgré sa très grande fragilité.

Pour reprendre la métaphore de la littérature, cela reviendrait à conserver dans un récit l'ensemble des phrases qui ont été élaborées avant de ne garder que la dernière qui sera imprimée.

Le Repentir (en italien *Pentimento*) dans l'Art est une forme bien identifiée de pratique qui consiste à transformer une première œuvre, jugée non satisfaisante par son auteur, en une nouvelle œuvre par superposition.

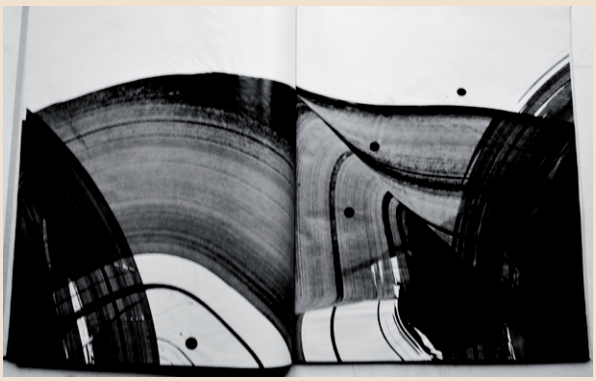
Nous ne sommes pas dans l'*encre-temps* dans la mesure où cette transformation n'intervient pas au moment de réaliser l'œuvre dans un processus d'interaction, comme nous avons pu le voir précédemment.
Ayant eu recours à ce principe à travers plusieurs techniques, il était intéressant d'en donner quelques exemples, pour enrichir, par différenciation, notre perception de l'*encre-temps*.

Nous allons parler de deux techniques parmi bien d'autres:

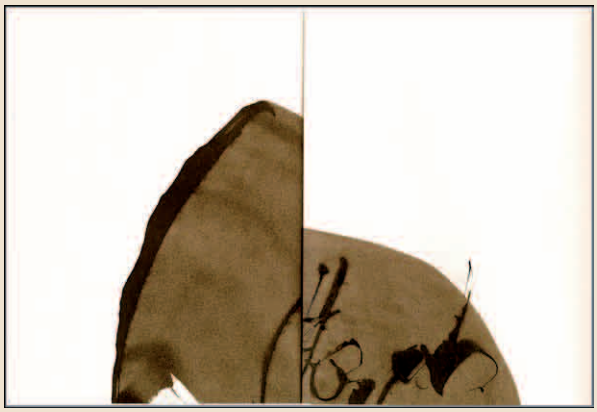
- la première fonctionne classiquement par une réintervention sur une trace déjà existante, afin de l'emmener vers autre chose. Une connaissance, qu'elle soit artistique ou scientifique, et une habileté aux techniques utilisées pourraient permettre au spectateur de reconstruire la chronologie des différentes interventions. C'est ce qui se passe lors de restaurations de tableaux. On retrouve le principe du déplacement des bateaux de Kant;
- la seconde technique est assez différente de la première puisqu'elle consiste à prendre une encre, réalisée généralement sur un grand format, et à opérer un pliage successif en deux, puis en deux… jusqu'à atteindre la dimension recherchée. Trois des quatres côtés sont coupés afin d'avoir accès aux parties pliées. Une redistribution des parties de la trace est ainsi faite, donnant une forme de distribution aléatoire des parties, recomposant plusieurs traces à partir d'une seule. L'aléa est ici contrôlé par la trace de départ qui, de part sa cohérence globale, rend les rencontres entre sous-parties pas complètement incongrues. Il arrive même qu'elles s'avèrent extrêmement fructueuses.

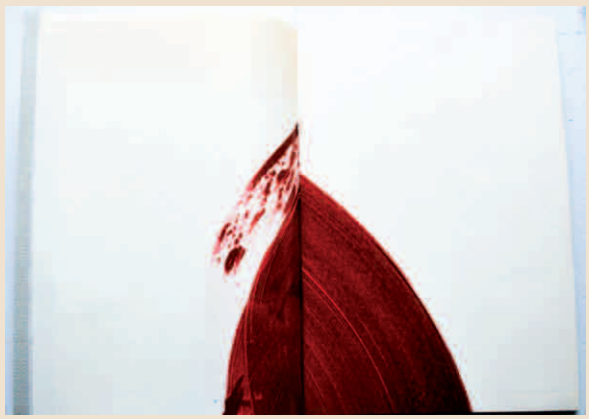
encre-temps structurel

Dans cette deuxième configuration (par pliage) l'interaction spatio-temporelle avec la trace n'est plus dans le médium lui-même, puisqu'inchangé lors du pliage, mais dans la structure de la trace, sa composition. En définitive un *encre-temps* structurel joue et apporte une richesse toute particulière au résultat, puisque la composition recomposée ne s'appuie pas sur le néant ou l'aléa total mais sur la structure de la trace globale initiale. Le télescopage spatio (pliage) - temporel (deux instants distincts) a bien lieu au niveau de la structure de la trace. On peut en effet parler d'*encre-temps* dans la mesure où il est impossible de remonter ni à la structure initiale, ni même à une chronologie temporelle à partir des structures fragmentaires et pourtant autonomes des sous-traces produites.







Temporalité asiatique et peinture

VIII-F

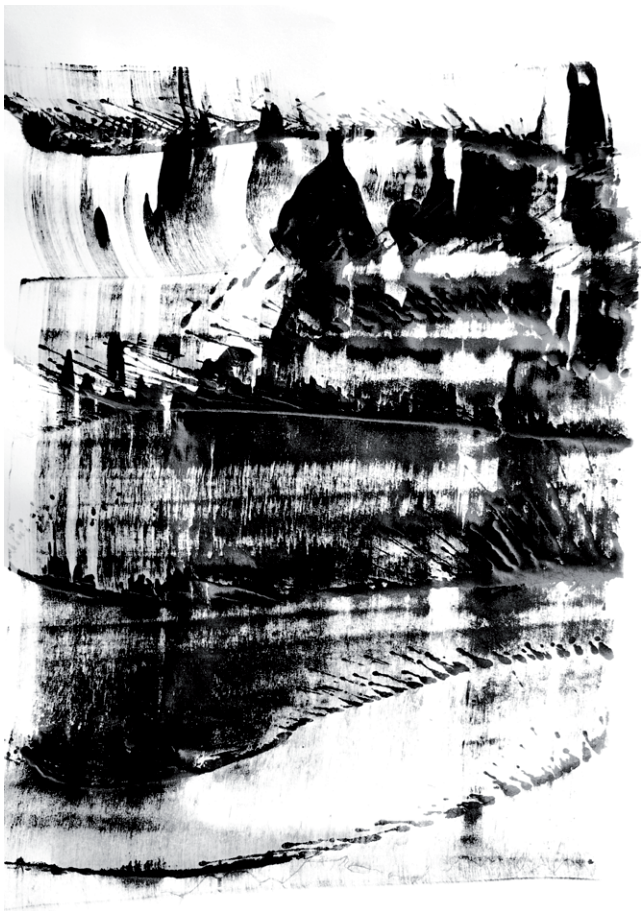
Éloge de la lumière
Pierre Soulages / Tanabe Chikuunsai IV
aux éditions 5 continents

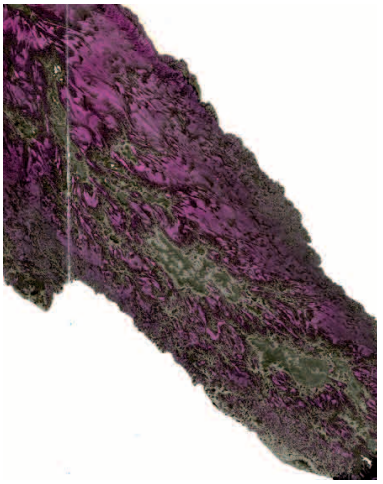
P. 13
«…Une des caractéristiques, à mon sens les plus fascinantes et les plus «exotique» de l'histoire de la peinture japonaise, est très certainement la technique picturale connue sous le nom de *ura-zaishiki* (les couleurs de l'envers) : propre à l'âge d'or de la peinture boudhique en particulier, elle consiste, afin de moduler les effets de la gamme chromatique, à appliquer certaines couleurs au revers du support de soie; apposés en couches plus ou moins épaisses, ces pigments, parfois mêlés de poudres métalliques, atténuent ou réhaussent ainsi le relief, l'intensité et l'éclat des motifs peints à même la surface de la soie.
Comme l'indique par ailleurs le maître contemporain de *nihonga*, Masaaki Miyasako, le caractère *ura*, dans la technique *ura-zaishiki*, qu'il a su interprêter sur des supports en papier de manière sensible et moderne, pourrait renvoyer également très anciennement par son étymologie à l'idéogramme du «cœur» ou de la «pensée», invisible, vibrant de l'intérieur; et voici donc qui confère, comme dans la peinture de Pierre Soulages, vers une intériorité spirituelle à ces couleurs cachées dont la présence et la luminosité se travaillent, s'unissent et se révèlent, du fond ou de l'envers, vers la surface, à travers le jeu de la matière…»

Laure Scwartz-Arenales,
Directrice de la fondation Baur à Genève

La temporalité dans la peinture asiatique est liée à la spatialité de la composition des scènes. La perspective est absente des scènes composant le tableau. Ces dernières sont sur des plans qui s'empilent verticalement.
En réalité, il s'agit davantage d'une organisation de l'espace où le peintre positionne différentes scènes sur un même plan, dont la lisibilité revient à la verticalité de la composition.
On peut néanmoins donner une dimension temporelle à cette organisation de la composition. En effet, le tableau se compose de différentes scènes qui pourraient être simultanées, comme on pourrait s'y attendre, mais qui pourraient également rendre compte d'une causalité temporelle (précédence) entre les évènements rapportés (le bateau de Kant).

La trace ci-contre adopte ce principe de temporalité intégrée à la représentation des différentes scènes du tableau. Il peut se lire de bas en haut avec à chaque fois la même scène (une prairie et un ciel à l'horizon) dont la temporalité (causalité) est marquée par l'évolution du ciel (de plus en plus nuageux).
Cette encre a été réalisée sur de la toile de verre épaisse (200 g/m^2) en utilisant, sans le savoir à l'époque, une technique japonaise ancienne (*ura-zaishiki*, cf. la note ci-contre) qui consistait à encrer le papier d'un côté de la feuille, puis de l'autre. J'ai utilisé l'eau pour diffuser l'encre.





VIII-G1) Conclusion

Nous avons mis l'accent, dans cet article, sur l'*encre-temps*, élément majeur dans l'action artistique en particulier, et peut-être dans l'action tout court.
Bien d'autres questions intéressantes dans le domaine de l'Art pourraient être évoquées. Par exemple on pourrait parler de l'émulsion (Larousse : «milieu hétérogène constitué par la dispersion, sous forme de fins globules, d'un liquide dans un autre liquide en phase continue») ou la rencontre du maigre et du gras, aporie s'il en est dans les manuels de pratique de peinture.
L'émulsion peut être considérée comme la rencontre, l'interaction et la fécondité portées par deux échelles temporelles pourtant a priori incompatibles entre elles.

J'ai mené des expériences dans ce sens en utilisant de la peinture à l'huile simultanément avec de l'encre de Chine. La miscibilité de l'huile avec l'eau est naturellement proche de zéro. Pourtant, toujours sous certaines conditions (par exemple en utilisant l'écrasement), leur émulsion produit des choses étonnantes, riches et inattendues.

En fonction des supports, l'*encre-temps* varie. Celle qui est liée à l'huile est quasiment infinie alors que celle liée à l'emploi de l'encre de Chine est de quelques secondes pour des épaisseurs relativement fines. L'émulsion de l'encre avec l'huile est une rencontre improbable où rien ne devrait fonctionner. Pourtant, Si l'on choisit une couleur à l'huile qui n'est pas très puissante (à faible tempérament d'absorption des autres couleurs), comme par exemple le rose, l'émulsion avec l'encre noire va produire des camaïeux de roses d'une délicatesse et d'une variété remarquables (cf. les encres ci-contre). Il faudra soigneusement éviter le rouge anglais (tout puissant) qui est capable d'avaler toutes les couleurs, même le noir, sans produire le moindre camaïeu.

L'*encre-temps* correspond en réalité à une durée pendant laquelle l'interaction avec la trace est possible. Dans le domaine artistique, on pourrait parler du temps de la création. Plus généralement, on pourrait parler du temps de l'action, qu'elle soit intellectuelle ou pratique.
On peut d'ailleurs relier ici l'*encre-temps* à une manifestation, une conséquence du principe de *dérive des connaissances*, vu dans la partie V-A1 du chapitre V. Si l'on regarde le côté intellectuel d'un travail quelconque, qui fait largement appel à la représentation des connaissances, on peut voir la dérive comme le mécanisme profond à la base de l'*encre-temps* correspondant.
Prenons l'exemple du romancier et parlons d'*écriture-temps* dans son cas. Cela correspond à sa faculté à revenir sur le temps d'écriture passé, au fur et à mesure de l'avancée de son roman. N'oublions pas que la dérive est un modèle cognitif de l'oubli qui pourrait ainsi définir les caractéristiques de cet *écriture-temps*. Jusqu'à quand et jusqu'où je peux réécrire sur ce qui a déjà été écrit sans tout reprendre ?
Je pourrais parler du programmeur, qui a exactement le même problème (cf. la partie X-A du chapitre X) que l'on pourrait nommer le *code-temps*…

L'*encre-temps* peut être généralisée comme étant une instanciation de l'épaisseur du temps. Cette épaisseur du temps complète les sept axiomes, proposés par Newton-Smith, et impliqués dans la topologie standard (les relations de précédence entre les instants).

Nous avons vu que cet *encre-temps* va de pair avec la notion de lenteur. En effet c'est la lenteur qui permet de l'exploiter efficacement. Cette dernière apporte des opportunités pour découvrir des espaces artistiques étonnants, mais cette condition, bien que nécessaire, n'est pas suffisante. C'est la combinaison des deux principes qui apporte toute la richesse que nous avons pu voir, très partiellement, au travers des exemples présentés en annexe dans cet article.

À l'époque où l'immédiateté est devenue la règle, pour ne pas dire la religion, il apparaît évident que le principe de la lenteur fait cruellement défaut. Non seulement cette immédiateté produit des choses qui échappent aux temporalités humaines, mais surtout, en réduisant le champ des possibles, elle ferme la porte à la diversité et déroule le tapis rouge à l'uniformité de pensée, d'action et de présence au monde.

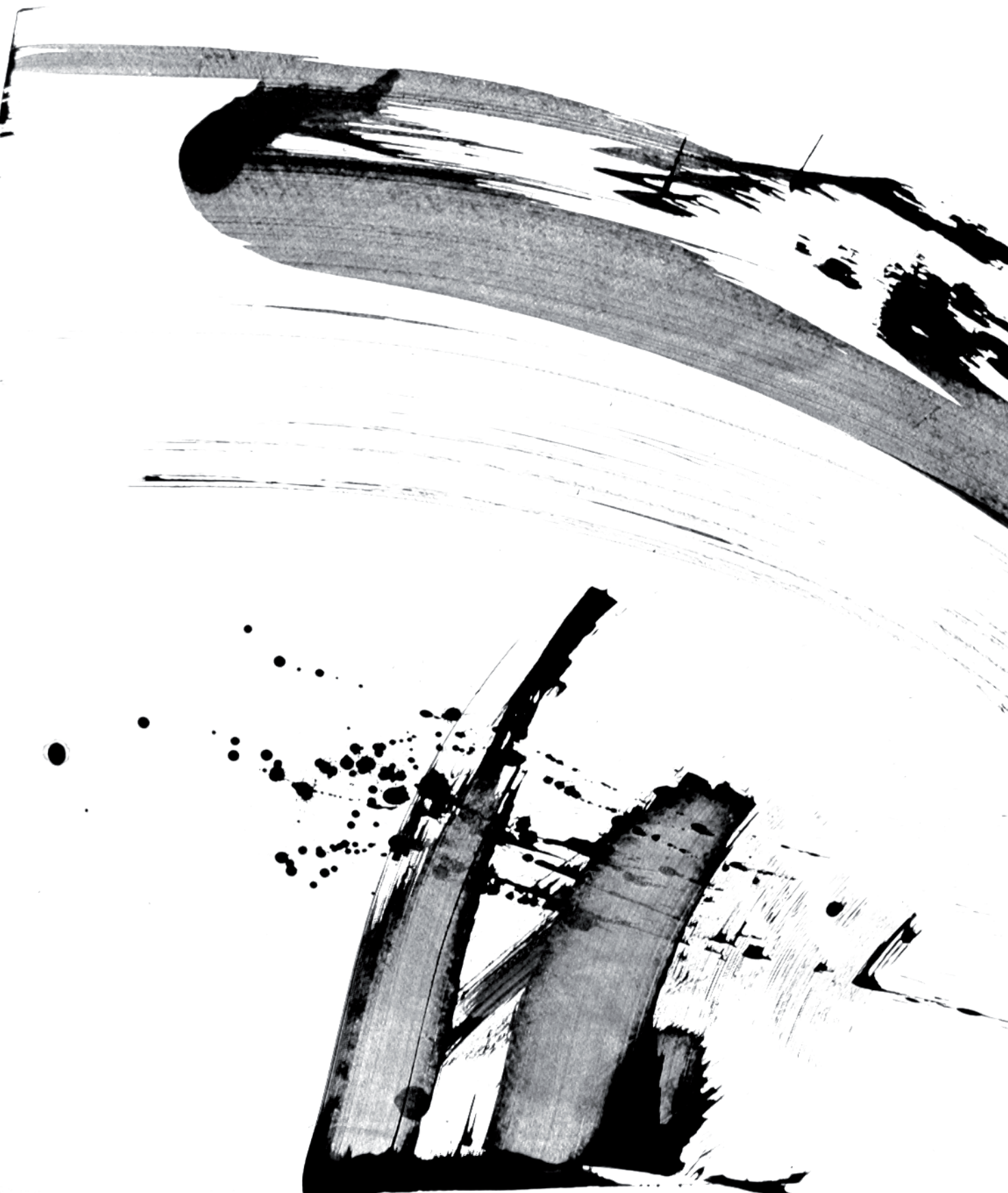
Je terminerai cet article par une expérience personnelle de faille temporelle que j'ai vécue grâce à une technique photographique du milieu du XIX^{e} siècle: le collodion humide[1].
Un photographe me fit mon portrait (cf. la photographie ci-contre) avec cette technique. Il fallait rester environ quatre minutes sans bouger devant l'objectif de la chambre photographique pour compenser le manque de lumière arrivant sur la plaque.
Ce fut un moment d'une rare intensité qui a paru durer une éternité, seul face à son image-reflet dans l'optique de la chambre. Un temps suspendu propice à l'introspection vagabonde.
Le calcul du temps de pause est complètement empirique, à l'image des sténopés (appareil photo sans optique, mais avec une simple fente dans une boîte et une surface sensible au point focal de cette fente).
Le résultat est marqué temporellement, au sens historique, par la technique, les visages sont durs, les traits tirés, la lumière subtile, renvoyant au XIX^{e} siècle et à l'image que l'on s'en fait aujourd'hui.

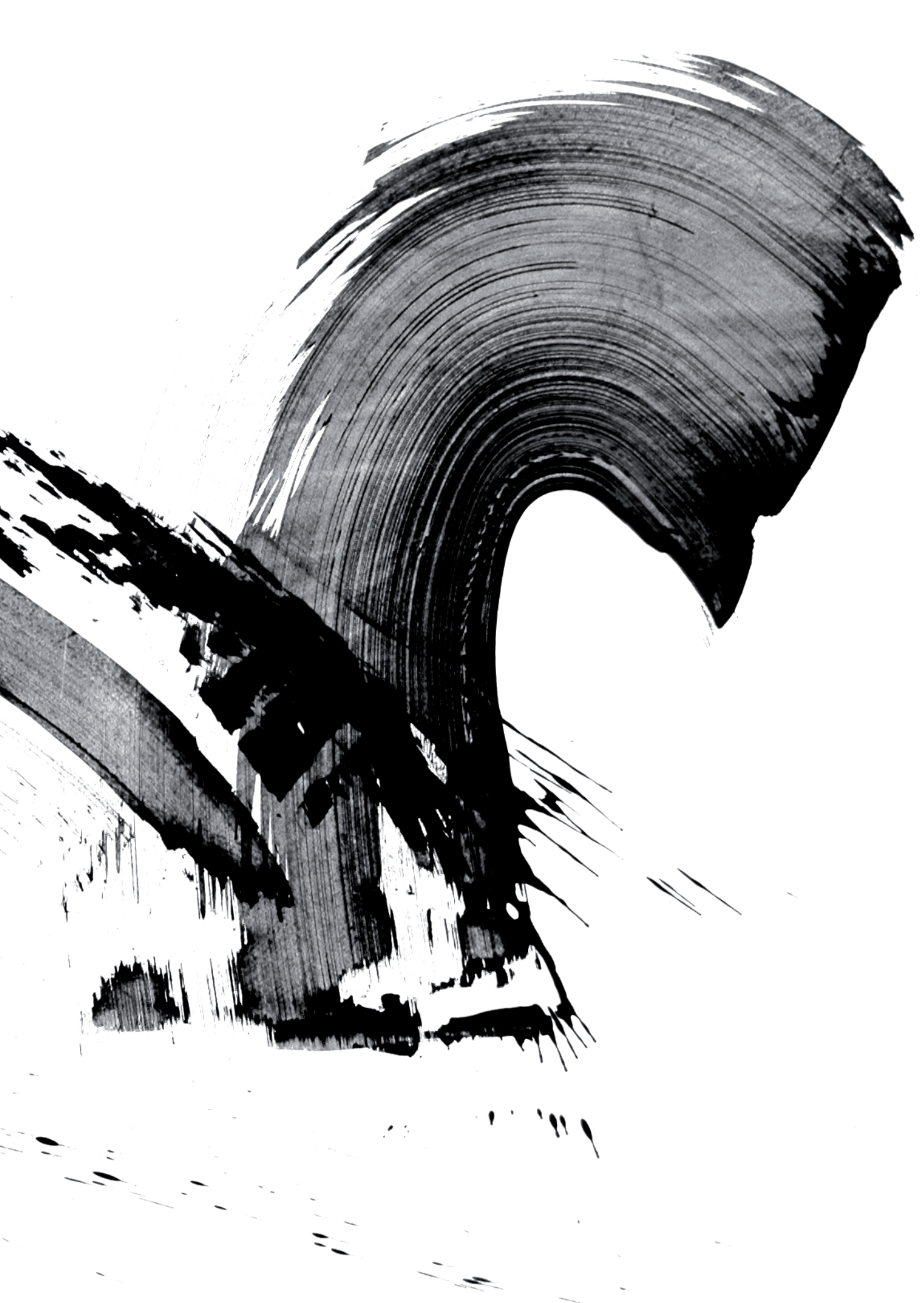
VIII-G2) Références

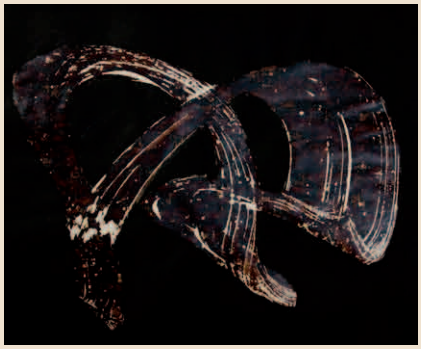
«La structure du temps: ontologie et représentation», Muriel Cahen, thèse de l'EHESS en Philosophie, soutenue le 04 novembre 2015.

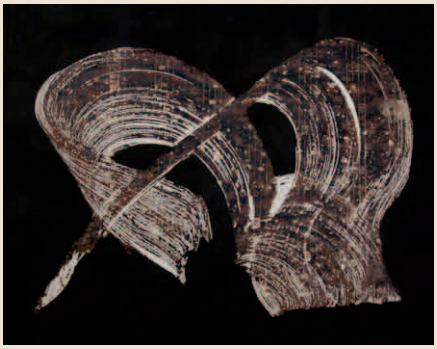
1 Le collodion humide est un procédé photographique négatif sur plaque de verre, où le collodion fixe l'émulsion sur la plaque. Il précède l'arrivée des négatifs au gélatino-bromure d'argent en 1880 (Wikipédia).

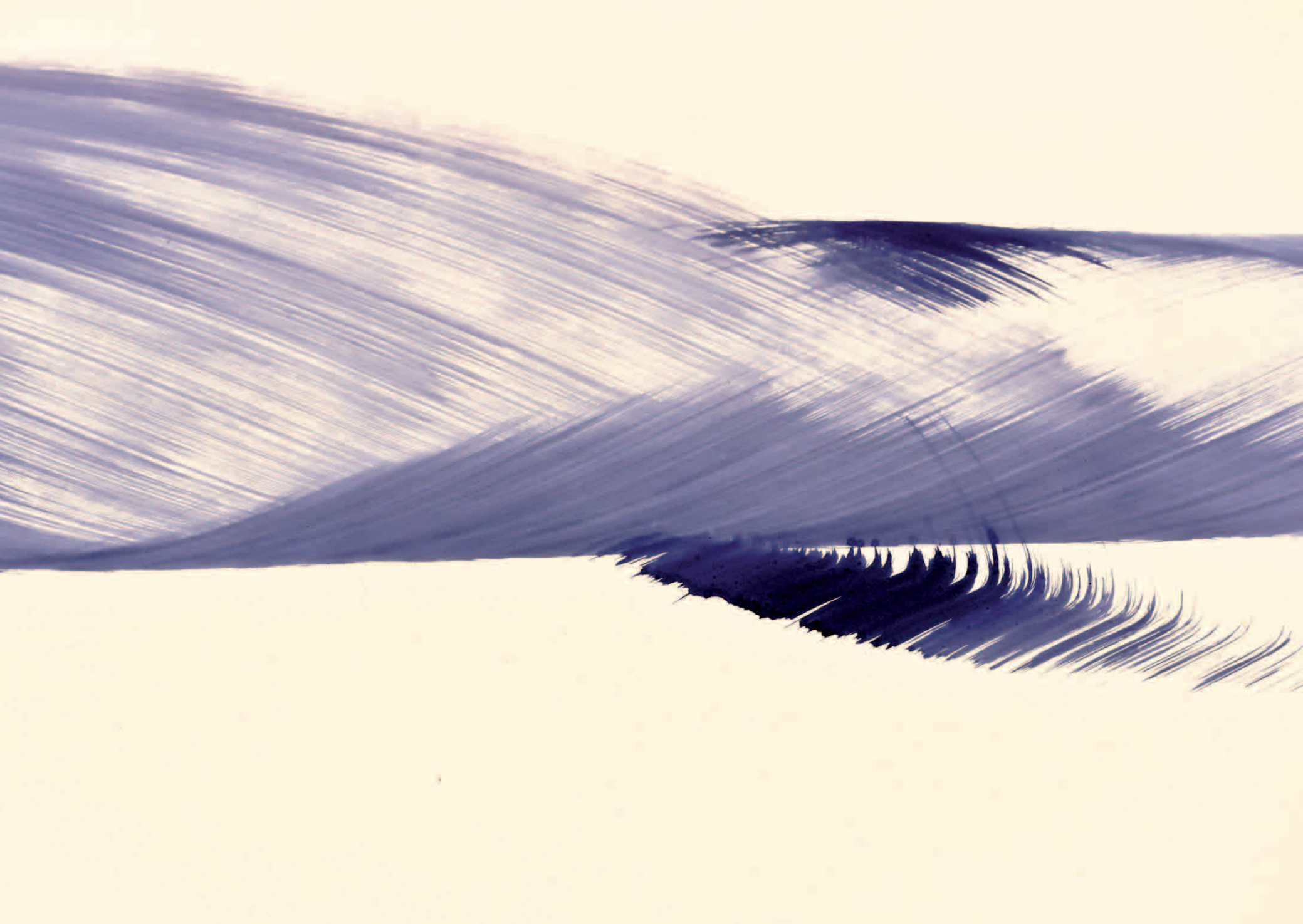
VIII-H1) ESPACES PRIMITIFS

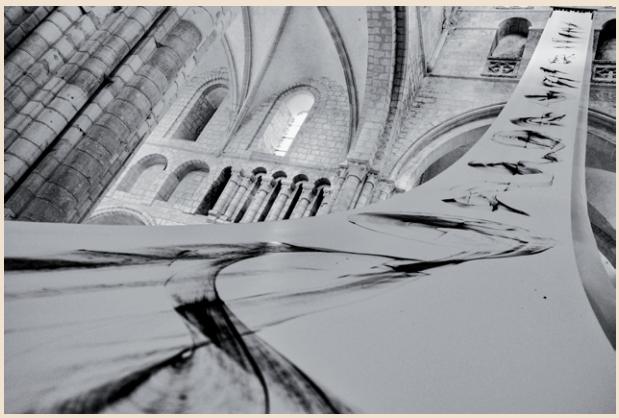


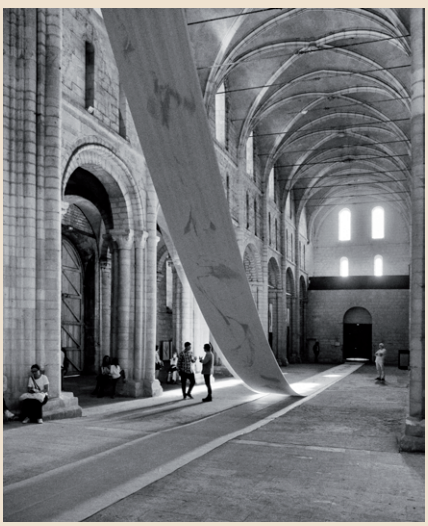


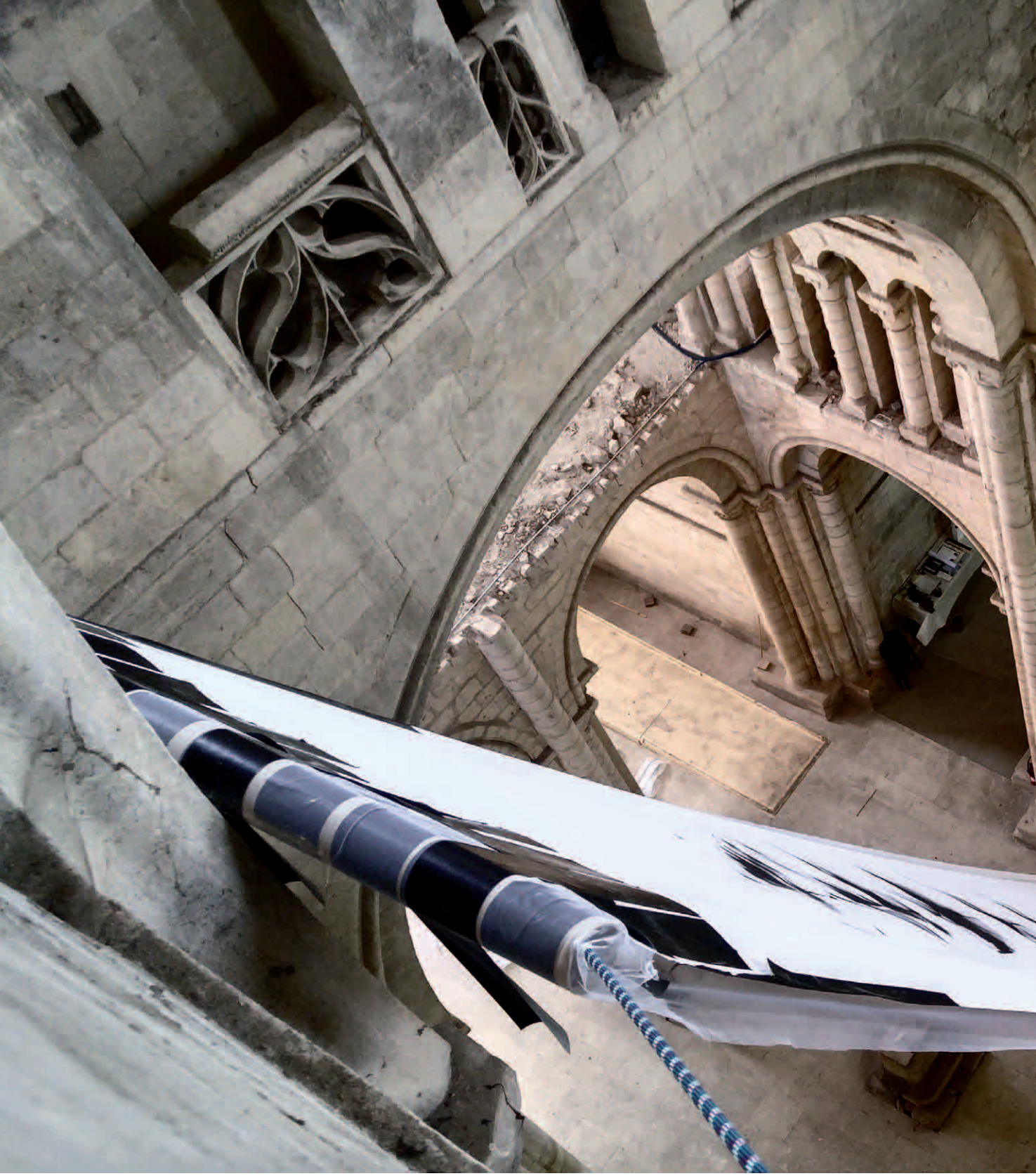


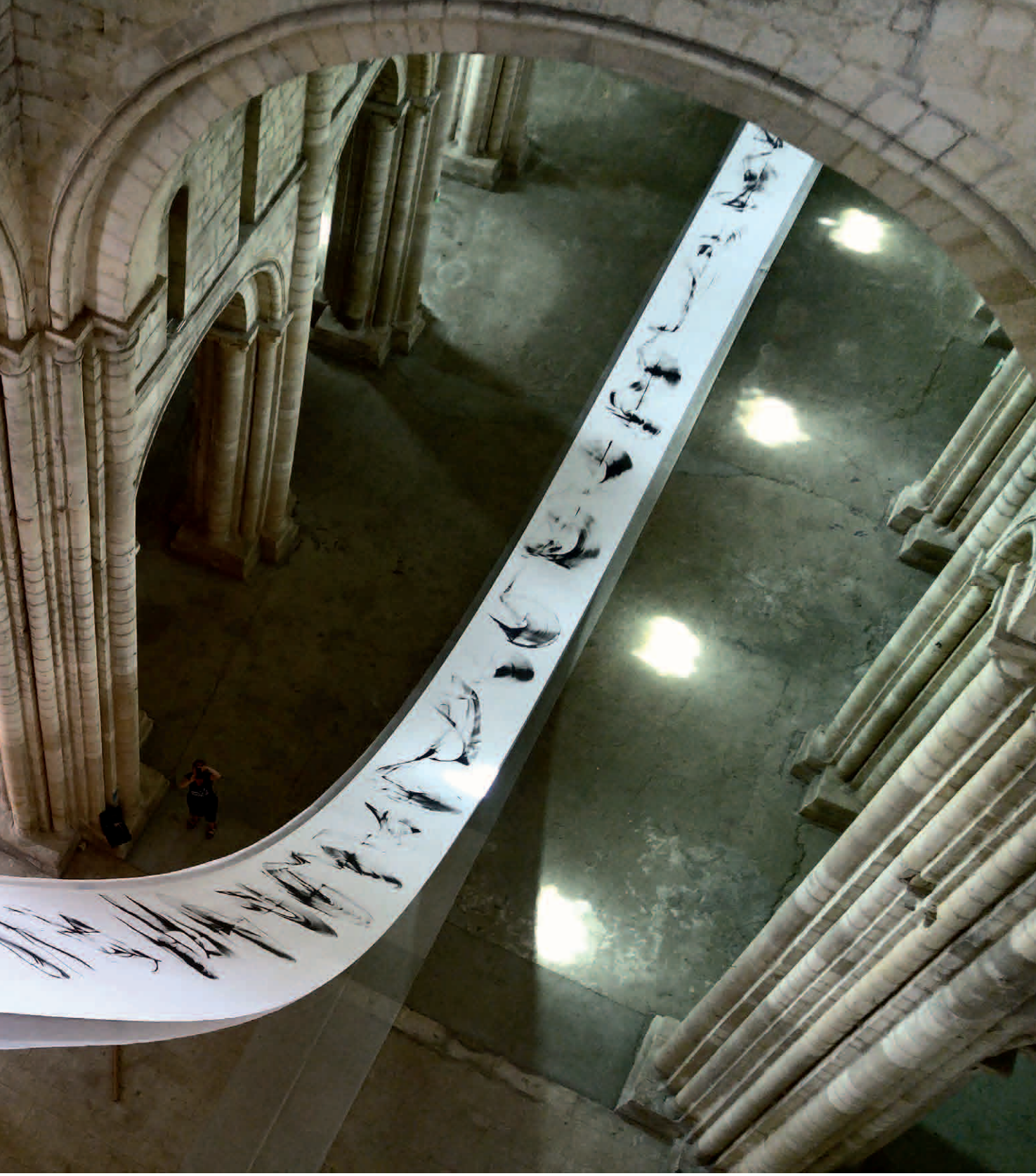


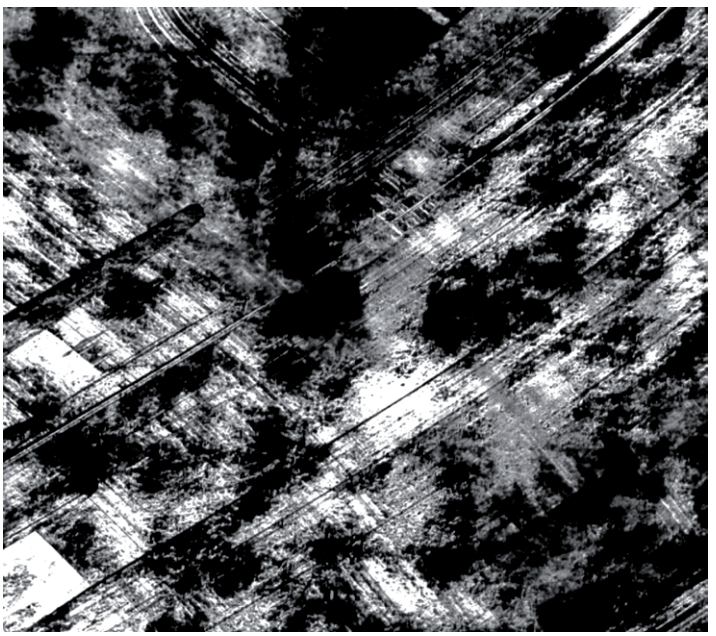


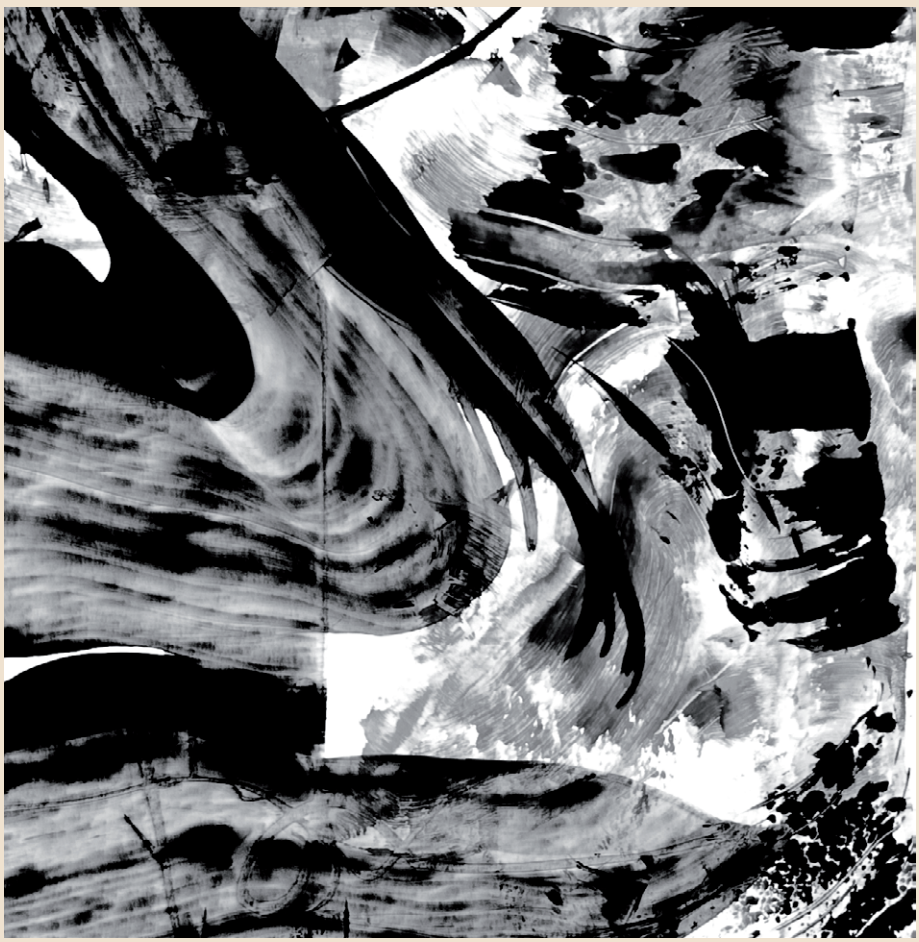


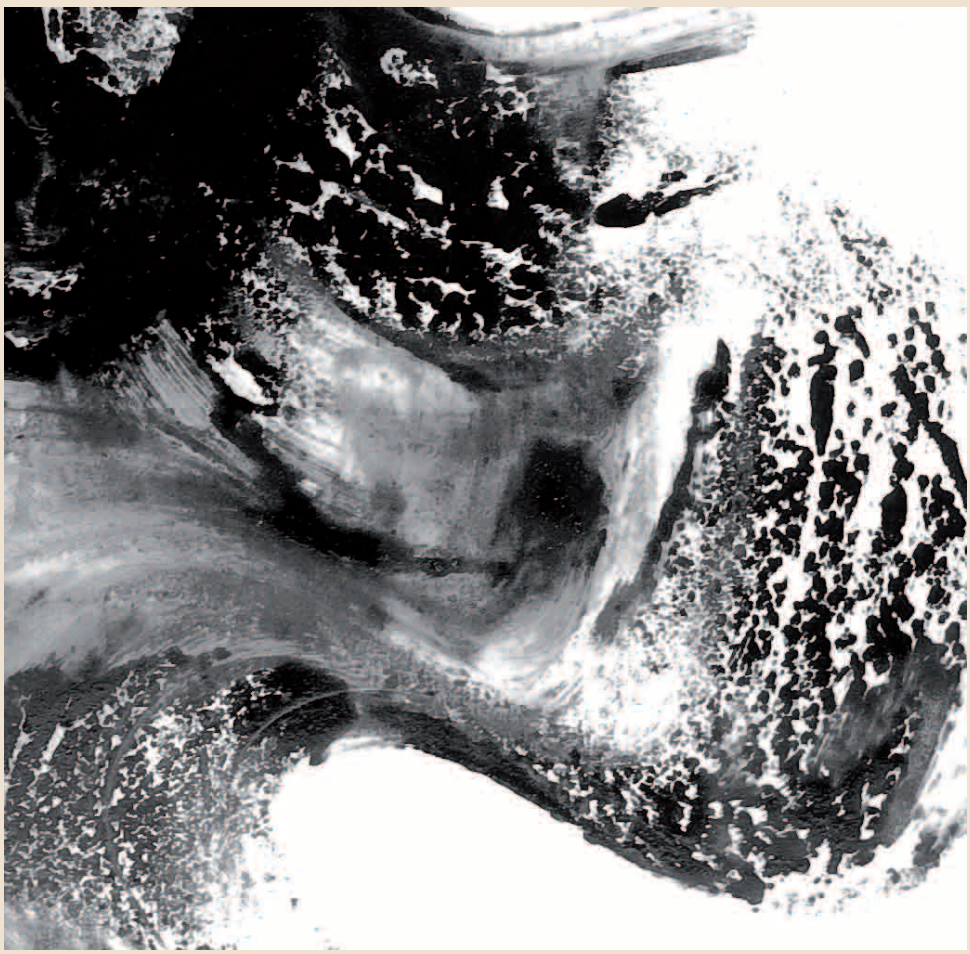


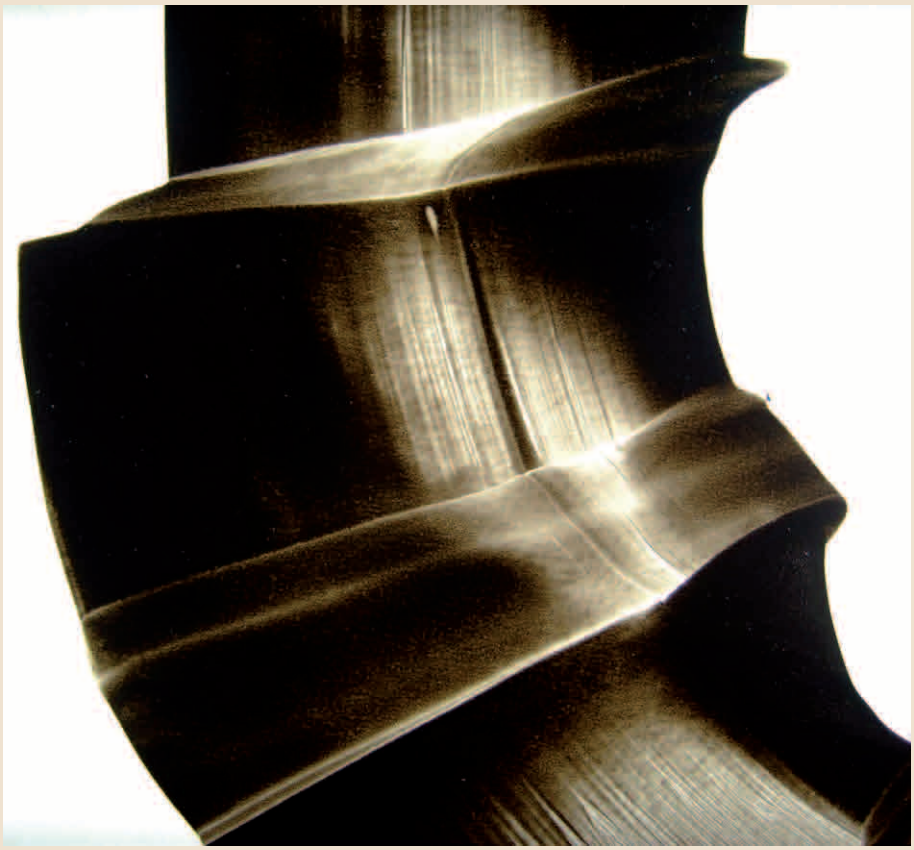


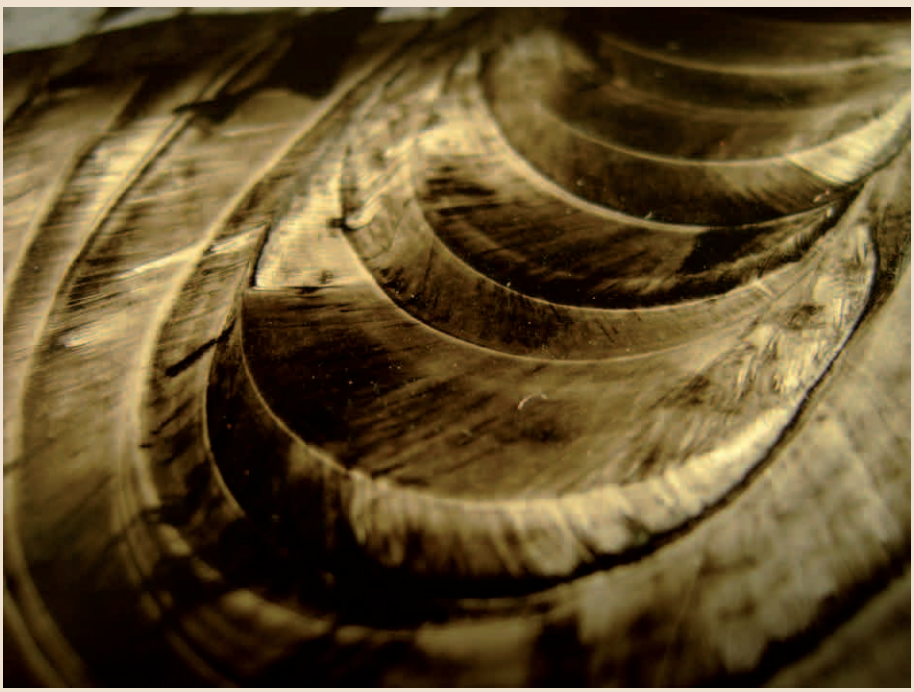


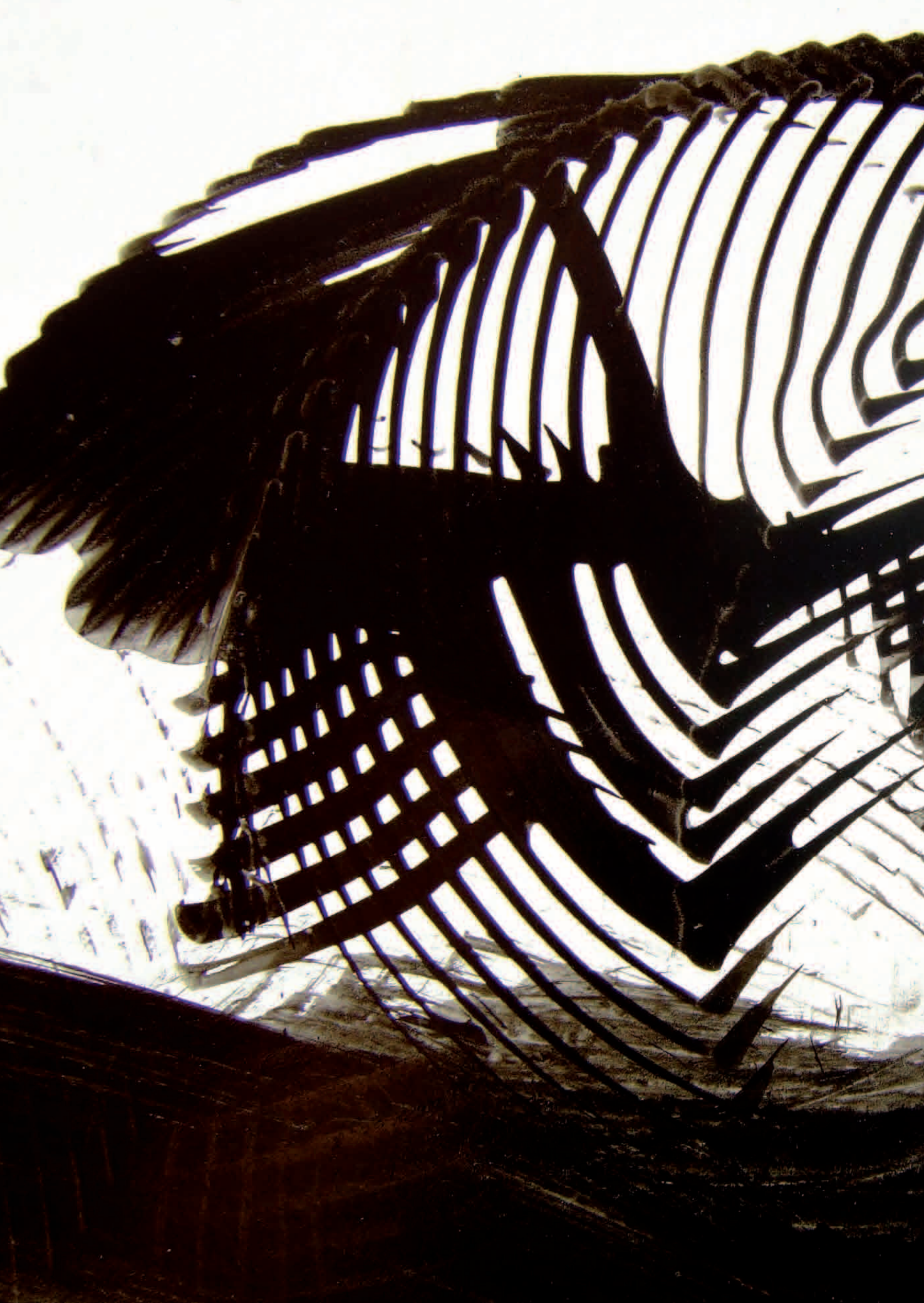


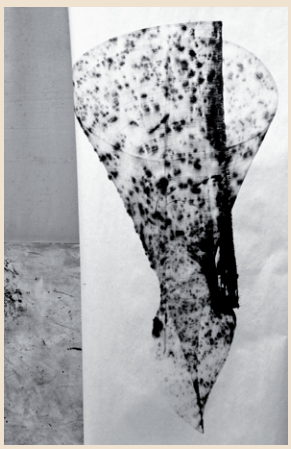


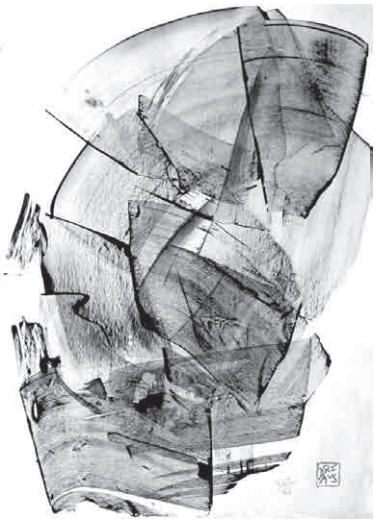


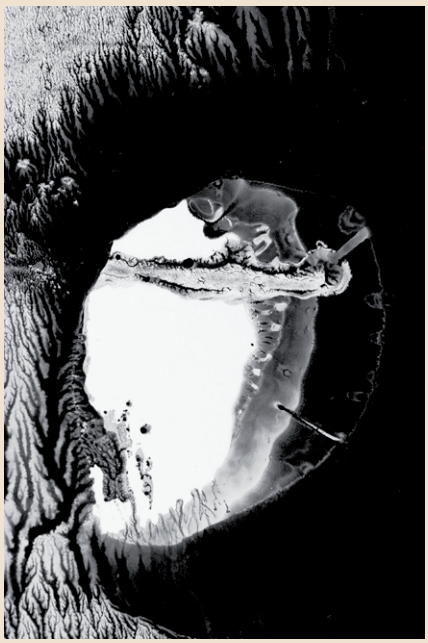


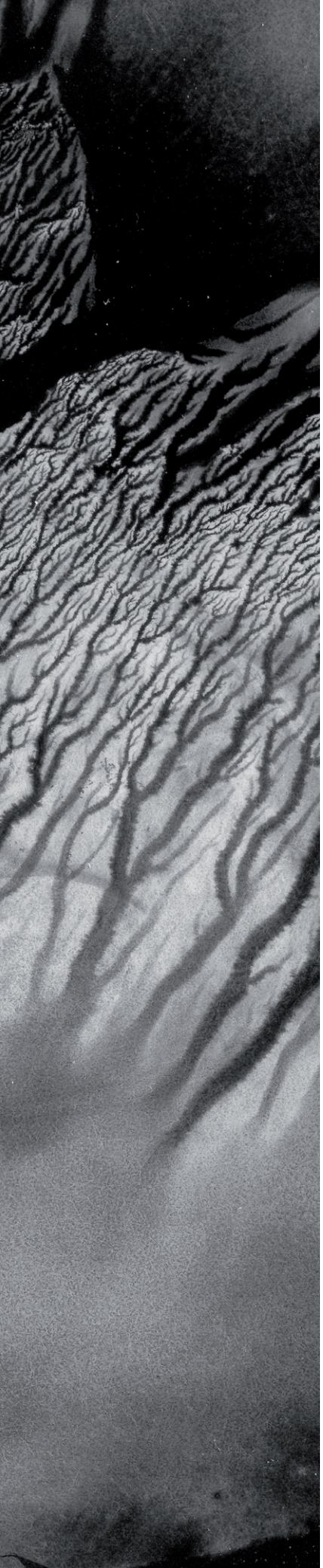


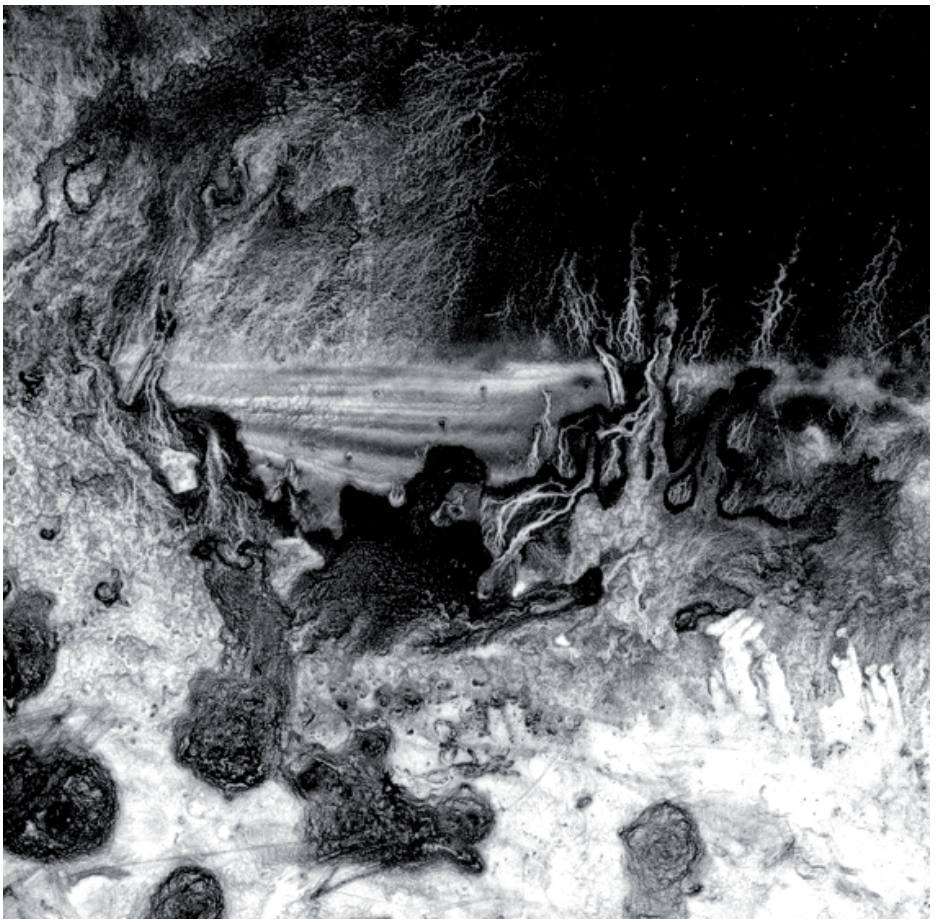


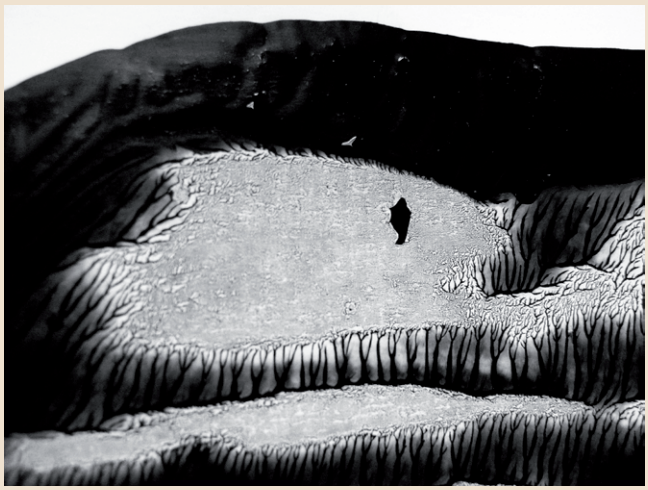


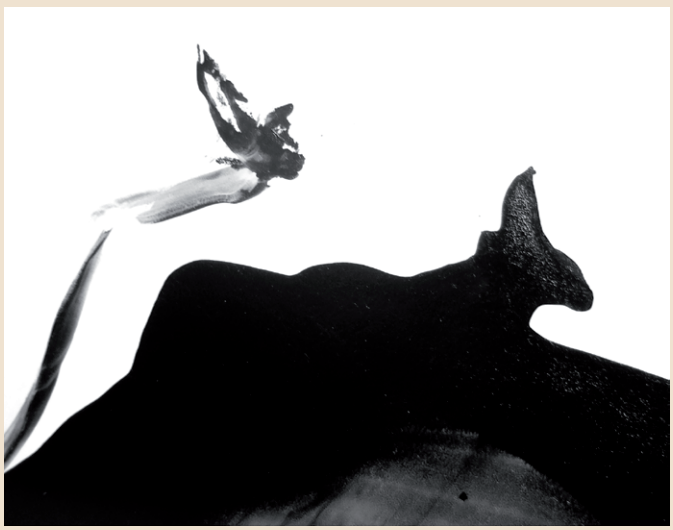


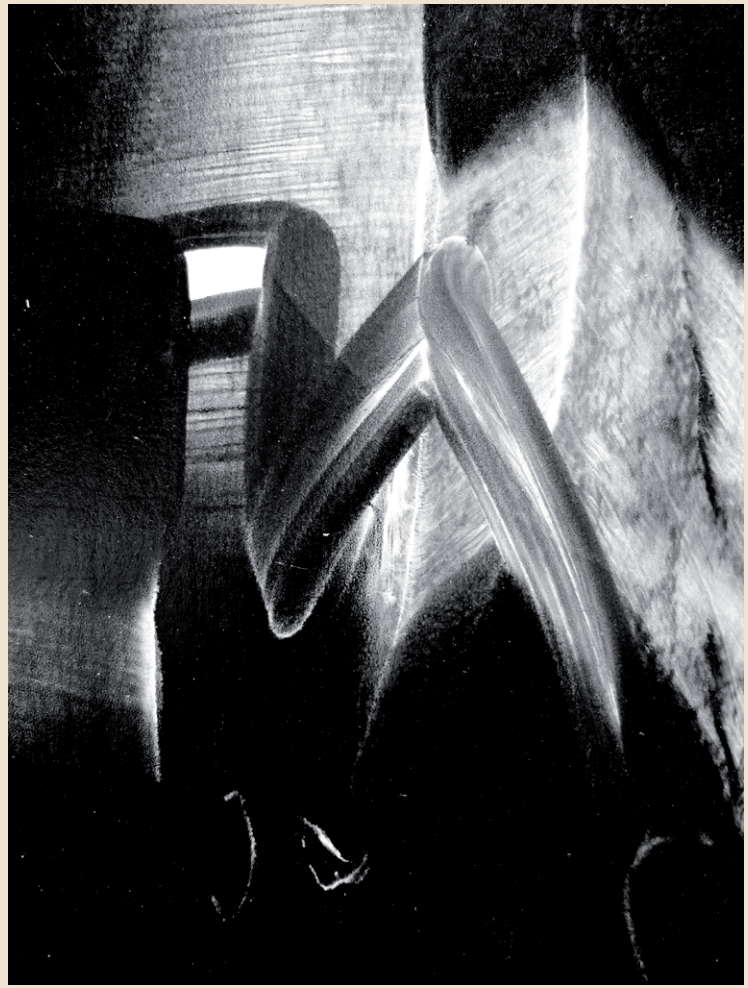


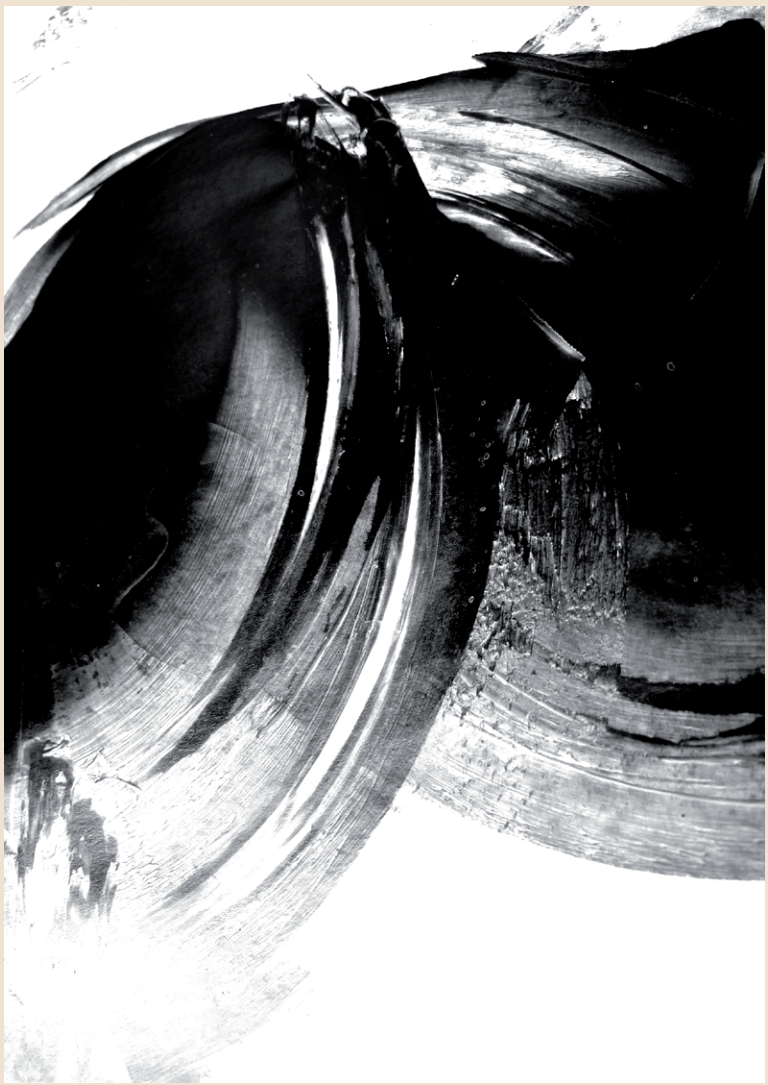


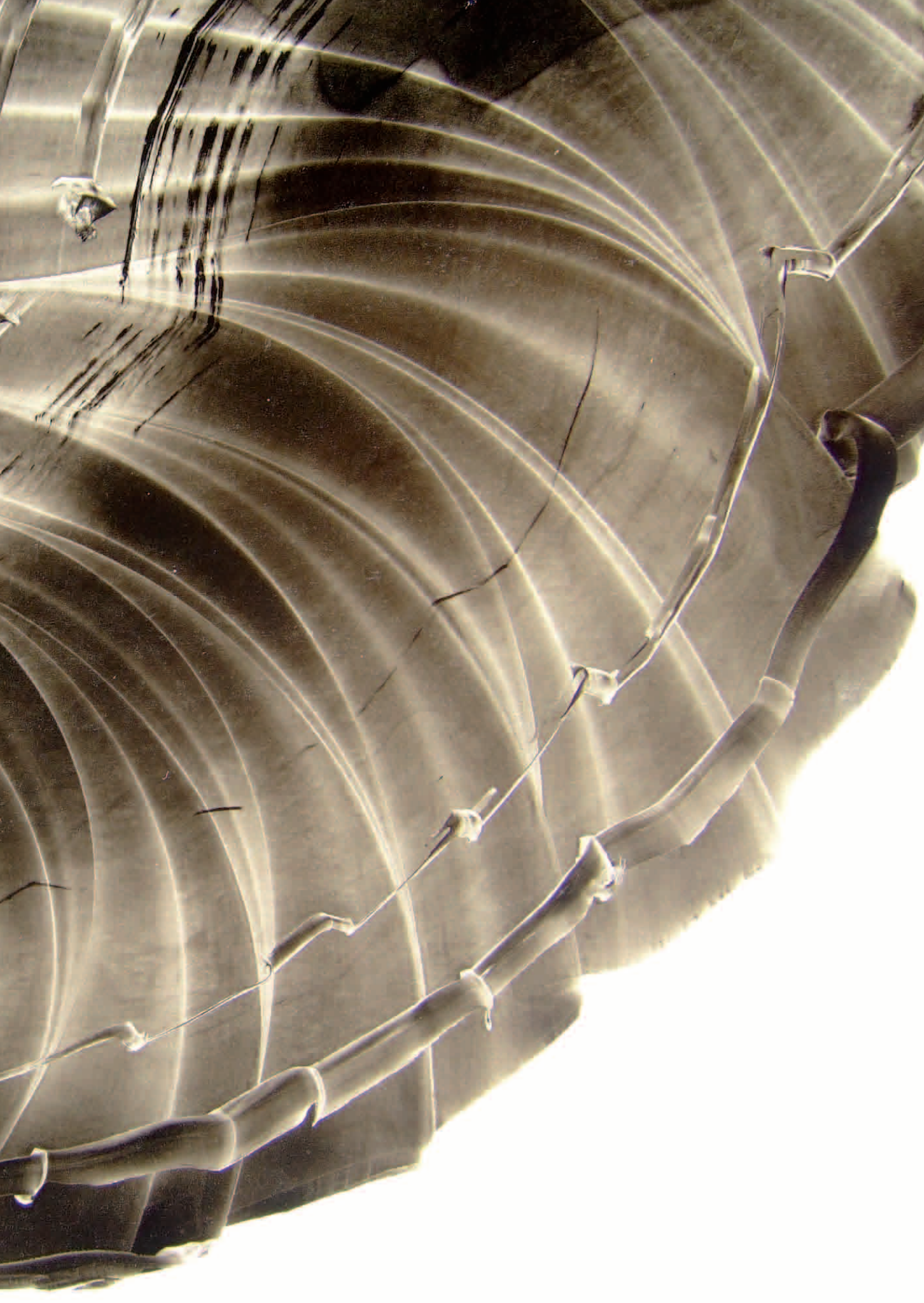


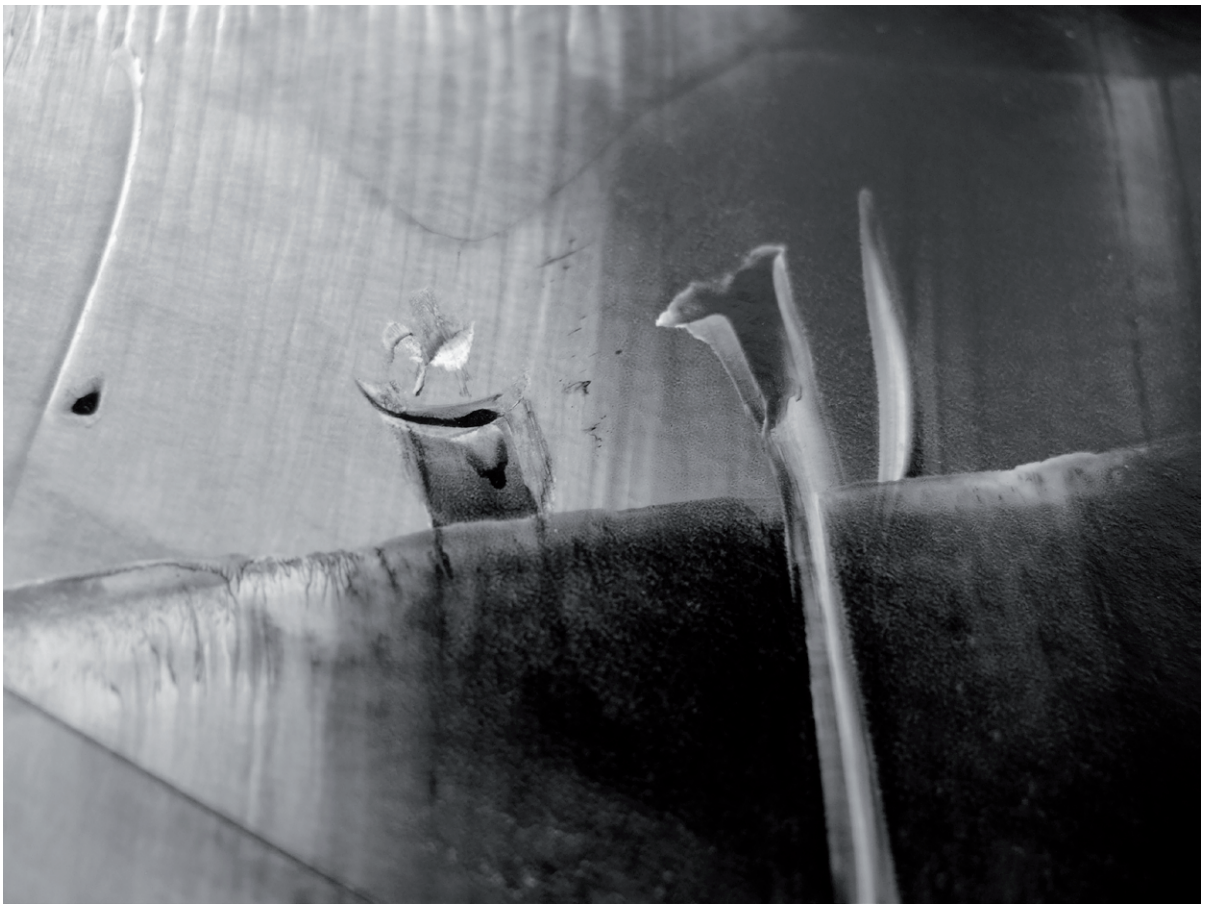


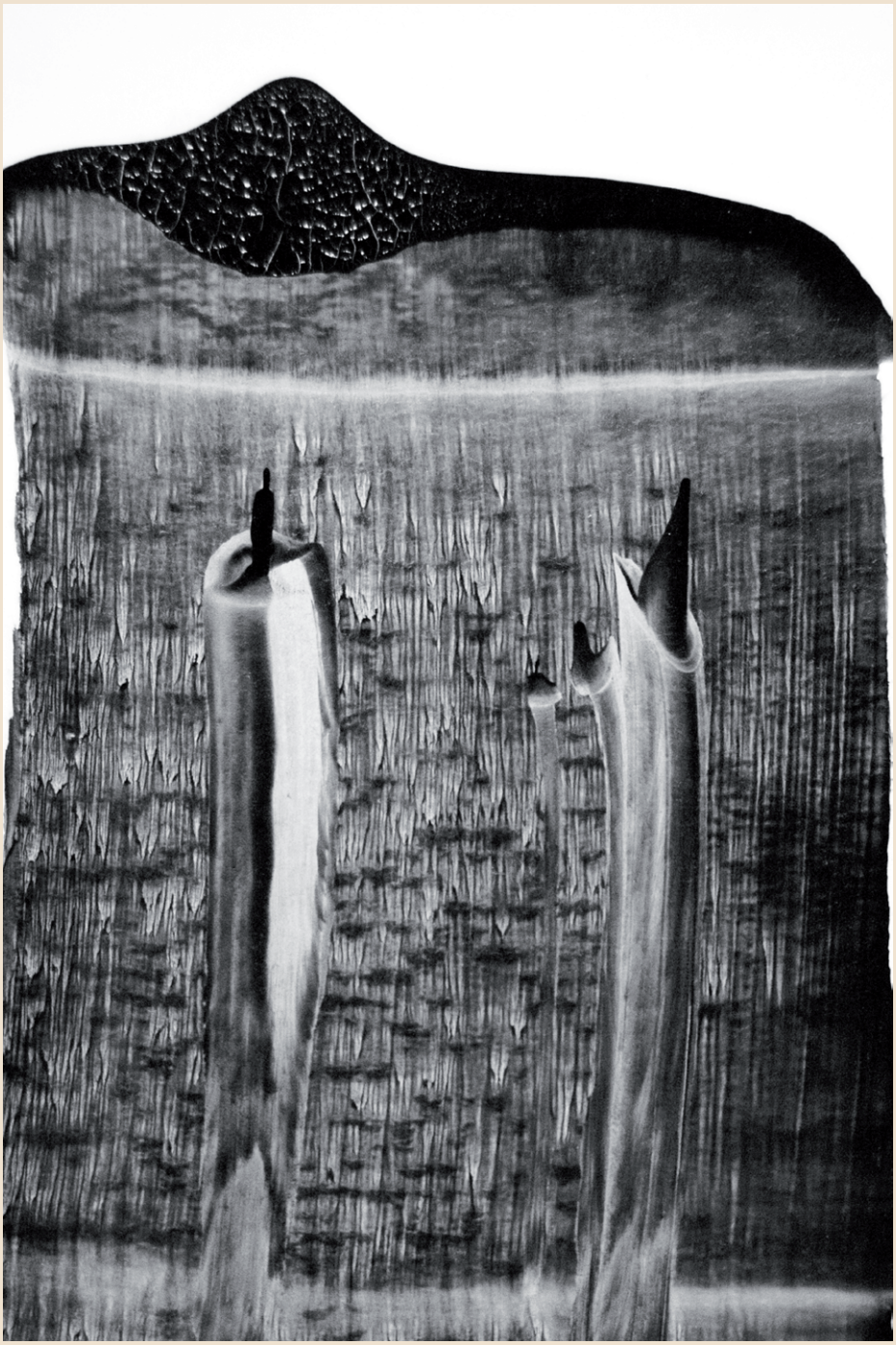


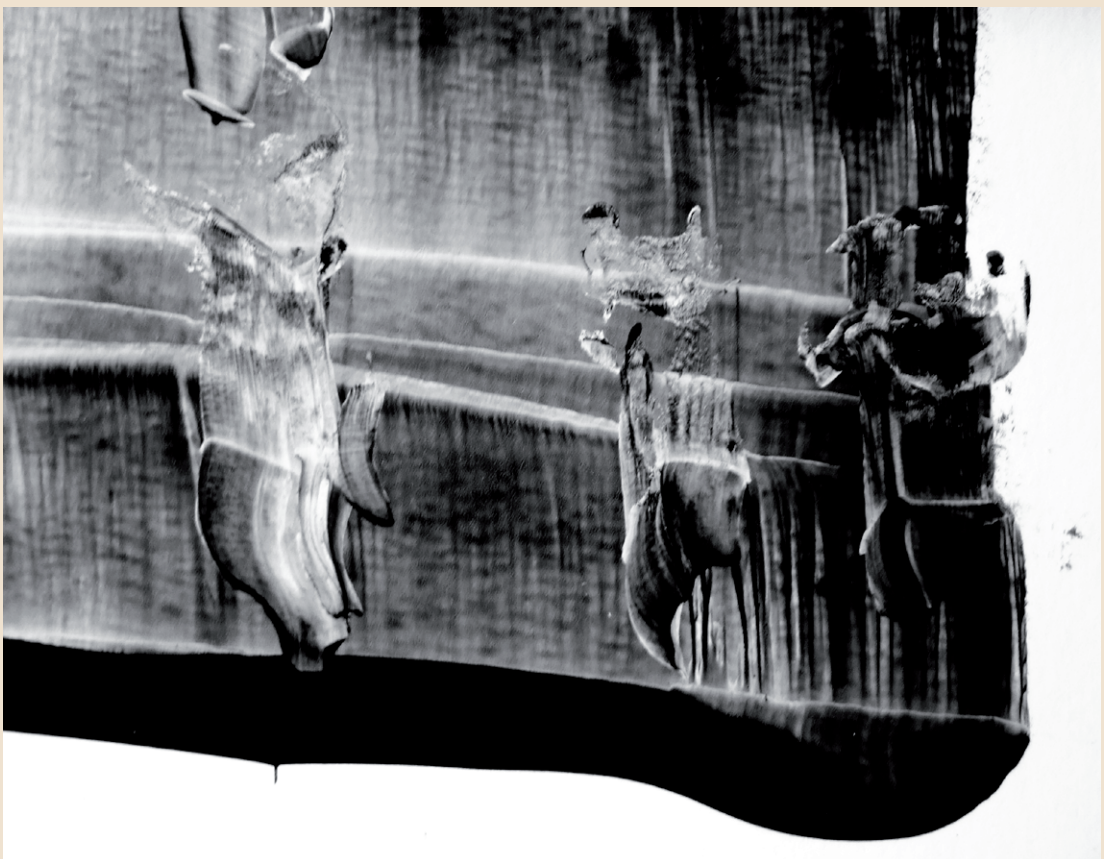


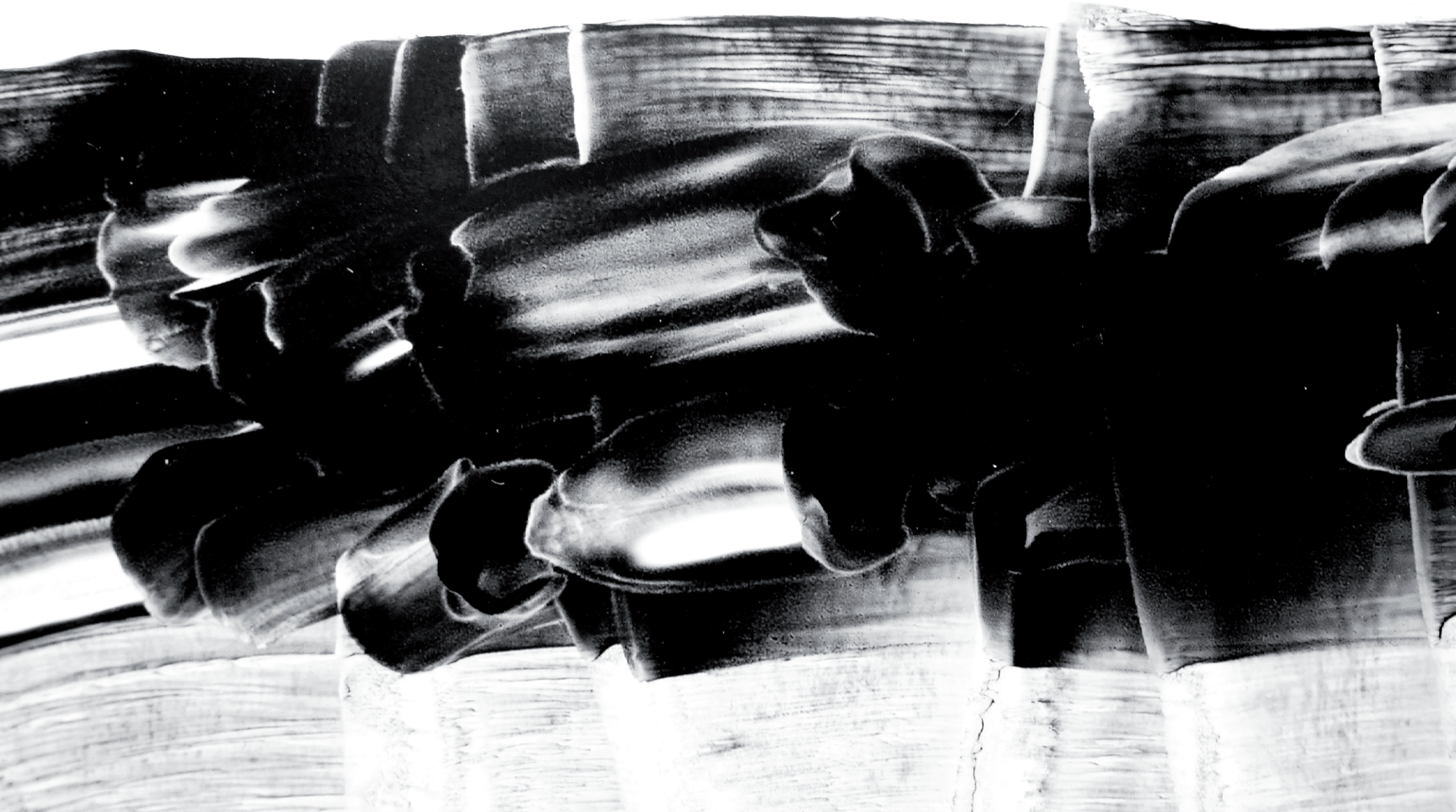


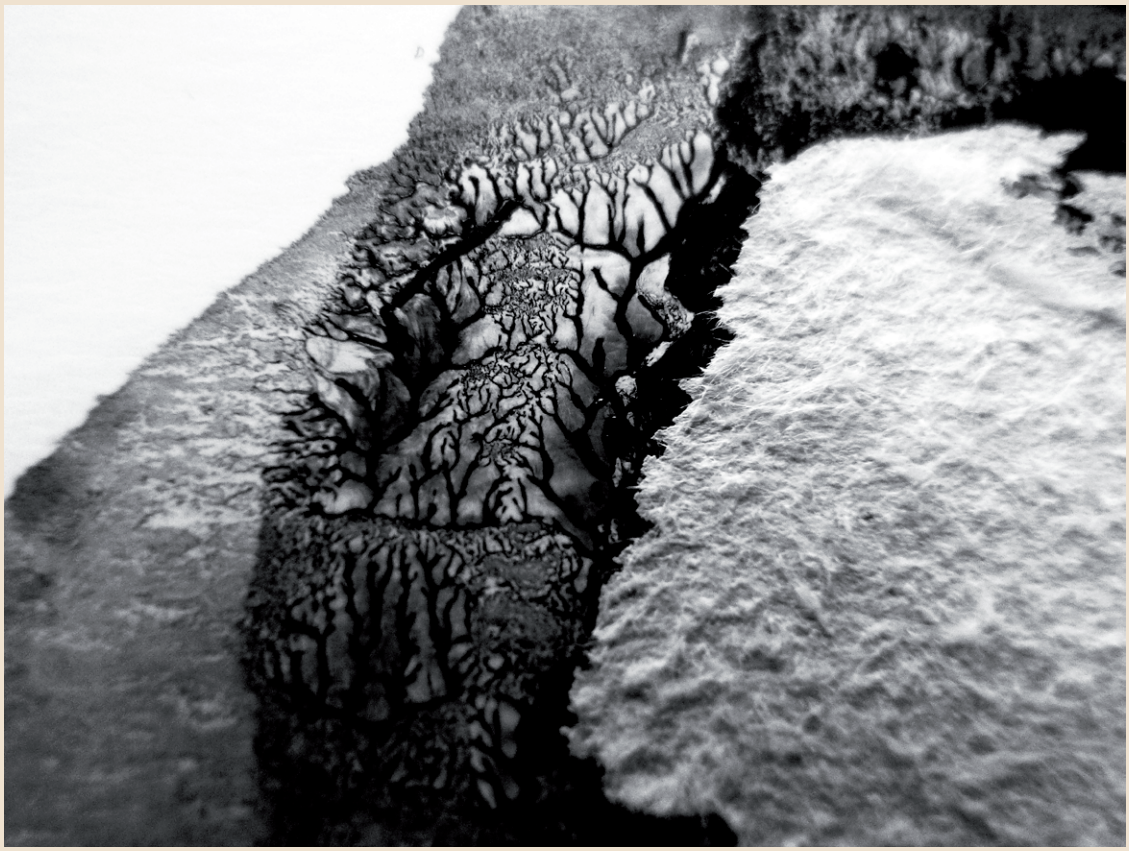


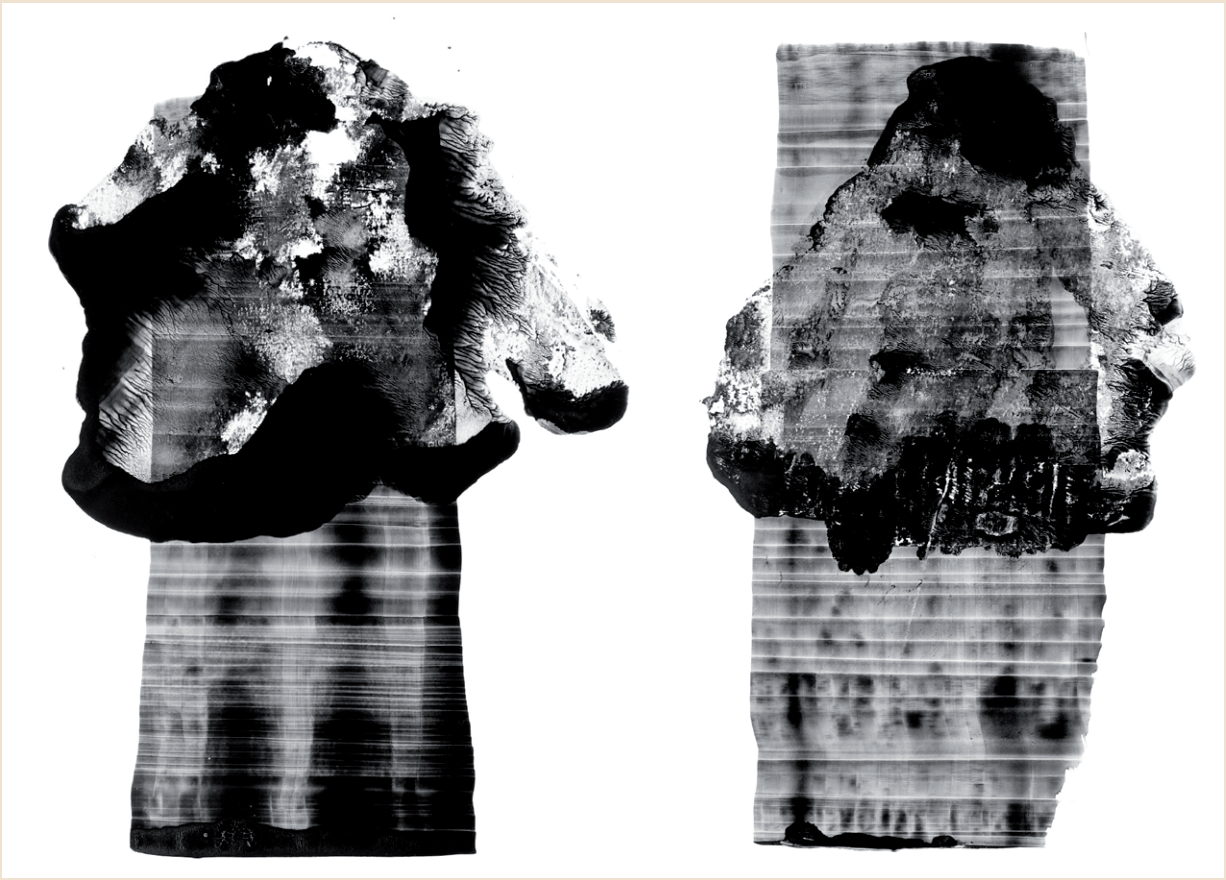


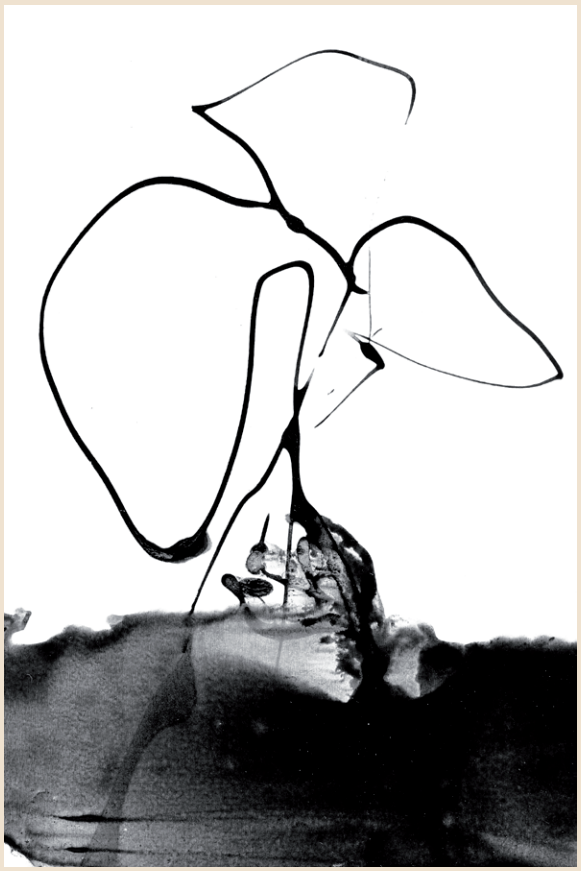


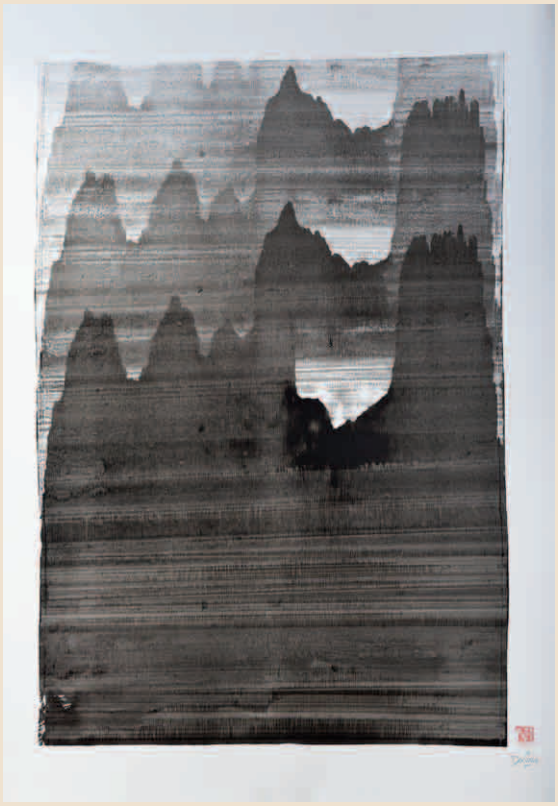


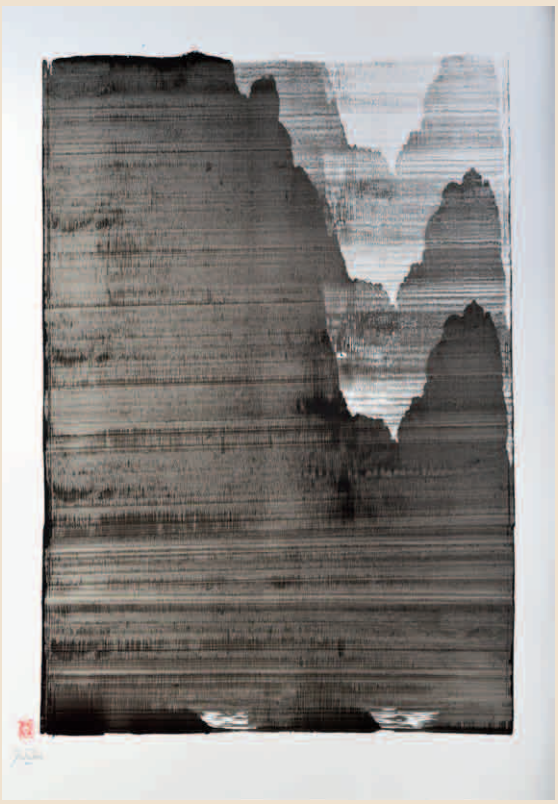












« […] Il en est d'autres que j'ai notés tandis que Picasso me les traduisait mot à mot de l'espagnol. Vu le genre d'intérêt extra-littéraire qu'ils présentent, il n'a pas paru utile d'obvier aux quelques inconvénients d'une telle traduction. On s'est même cru autorisé à violenter la syntaxe française chaque fois qu'elle se refusait à suivre l'espagnole plutôt que de se priver d'un tour expressif et de compromettre par là le mouvement de la pensée » (Cahiers d'art, 1936, p. 52)[1].

1 « *André Breton explique en effet que ces traductions ont été faites oralement sous la dictée de Picasso* », dans « *Pablo Picasso. Écrits 1935-1959* », p. 16, aux éditions Gallimard dans la collection Quarto.

TEXTES OU LA CHRONOLOGIE D'UNE PENSÉE

IX

Un testament de recherche, tel que je l'ai imaginé, rend compte d'un travail au long court qui aura été à la base de mes activités de recherche pendant ma carrière de chercheur au S.E.P.T. à Caen, puis d'enseignant-chercheur à l'Université de Caen. Je ne pouvais pas ne pas évoquer toutes ces années d'errance sans donner quelques éléments chronologiques. La rédaction de ma thèse de 1989 à 1992, puis de mon Habilitation à Diriger des Recherches (HDR) de 1997 à janvier 1998, furent évidemment des jalons importants dans cette chronologie. D'ailleurs l'exercice de rédaction de mon testament (incluant les développements de codes), qui dura de septembre 2019 à octobre 2023, en fut une troisième occurrence tout aussi marquante que les deux premières, mais terminale !

Pour alimenter ces documents, qui représentent autant de facettes différentes de la face visible de l'iceberg, il fallait bien travailler la matière en ressassant tant et bien des idées où le brouillard était de mise.

Pour donner un petit aperçu, mais réaliste, de ce paysage fantasque et possiblement décourageant, dans lequel je me suis aventuré, j'ai souhaité exhumer quelques bouts de griffonnages, écrits à la main et datés.

Introduction

IX-A

Avant d'apréhender ces fragments de réflexions, je dois dire quelques mots que m'ont inspiré les trois jalons marquants, auxquels j'ai fait référence un peu plus haut; la rédaction du testament en étant le plus prégnant des trois.

Je veux parler du rapport entre le fond et la forme du texte d'un côté, et son auteur de l'autre.
La recherche est une activité qui peut encourager la souplesse et l'approximation, afin de ne pas se fermer de possibilités. Vu de loin cela pourrait s'apparenter à un mouvement brownien au pays des idées et des intuitions, c.-à-d., d'un certain point de vue, sur le fond!

Cela se voit par les nombreux allers-retours et les imprécisions aventureuses/audacieuses/inutiles dans le voyage temporel des textes produits, traduisant l'épaisseur du brouillard ambiant.

L'écriture du testament nécessite un récit linéaire, formant une histoire et demandant une très grande rigueur sur le fond (chronologie des concepts, clarification, démonstrations et preuves…), mais aussi sur la forme (typographie, choix du système de notation…).

J'ai choisi de rédiger mon récit en avançant conjointement sur le fond et sur la forme, créant malgré moi, un rapport de force entre la forme, Elle, et moi, l'auteur, dans lequel Elle avait de plus en plus de poids.
L'apogée de ce combat entre Elle et ma nature profonde, lui donnant une victoire écrasante, fut lors de la fin de la rédaction. Le coup de grâce me fut asséné lors des nombreuses relectures papier de l'ensemble du texte.

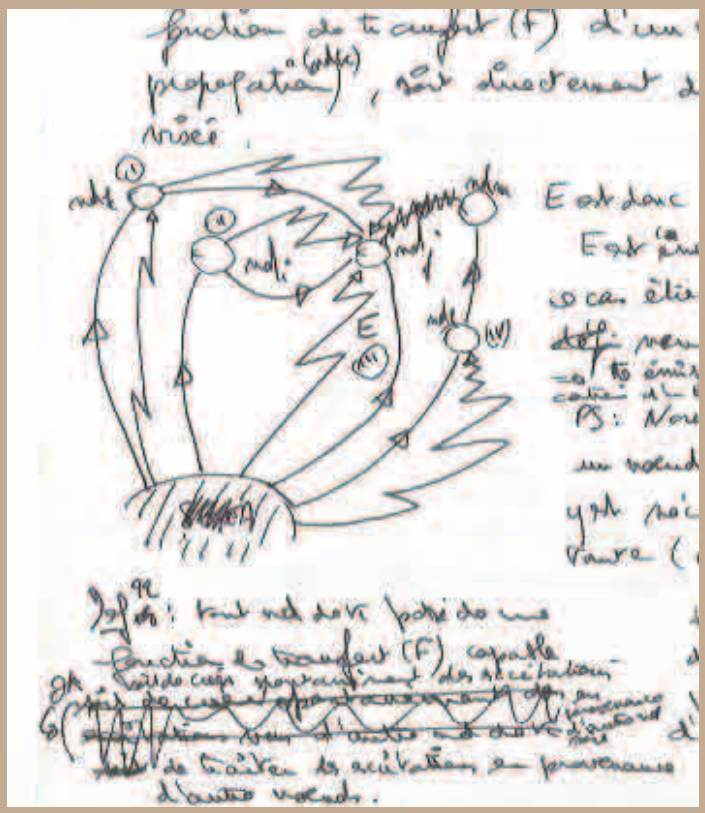
La forme m'imposa de coucher sur le papier l'anti-sèche des contraintes typologiques (par exemple mettre tel espace avant les signes de ponctuation, choisir le système de taille de police de caractères et celui des interlignes pour les titres et les paragraphes, quoi mettre en italique…) qu'il fallait appliquer, de façon cohérente, à la totalité du texte.

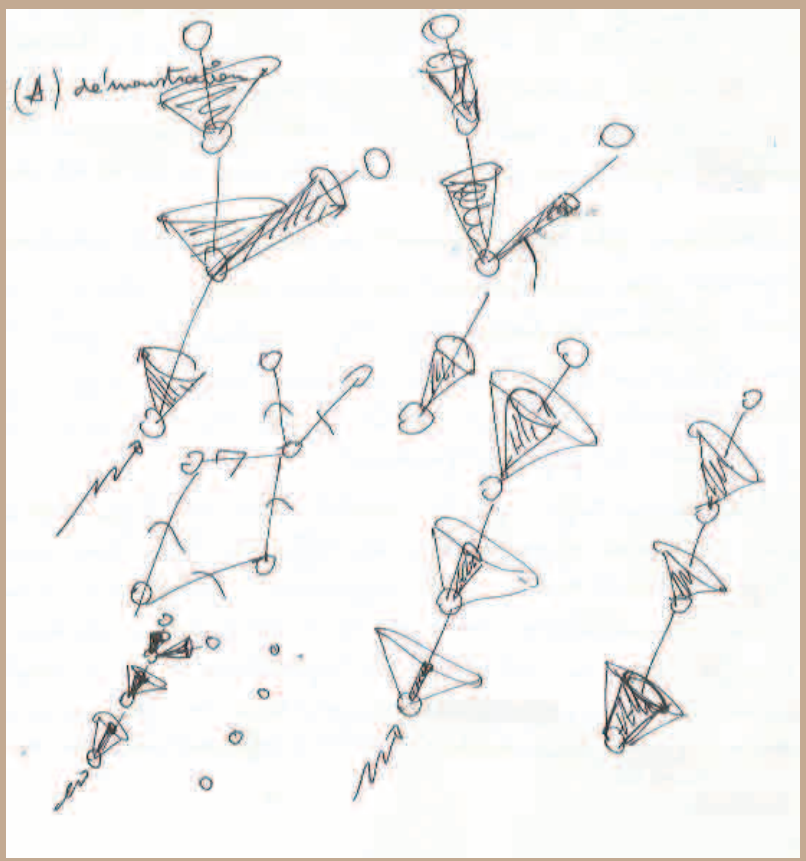
Il est évident qu'Elle a joué un rôle décisif dans l'intelligibilité du texte. Le fait d'avoir fait appel à Elle dès le début de la rédaction, introduisit une forme de circularité généreuse et féconde entre Elle et le récit lui-même.

Le report, dans la version numérique, des modifications relevées lors des relectures papier, fut très intense. Il me permit, à l'image du modèle de la *dérive des connaissances*, d'intégrer/vérifier une «dernière» fois et simultanément, presque toutes les contraintes typographiques que je me suis vu fixer par cette circularité fond/forme. Ce que j'étais incapable de faire (prendre en compte simultanément toutes ces contraintes) tout au long de la rédaction.

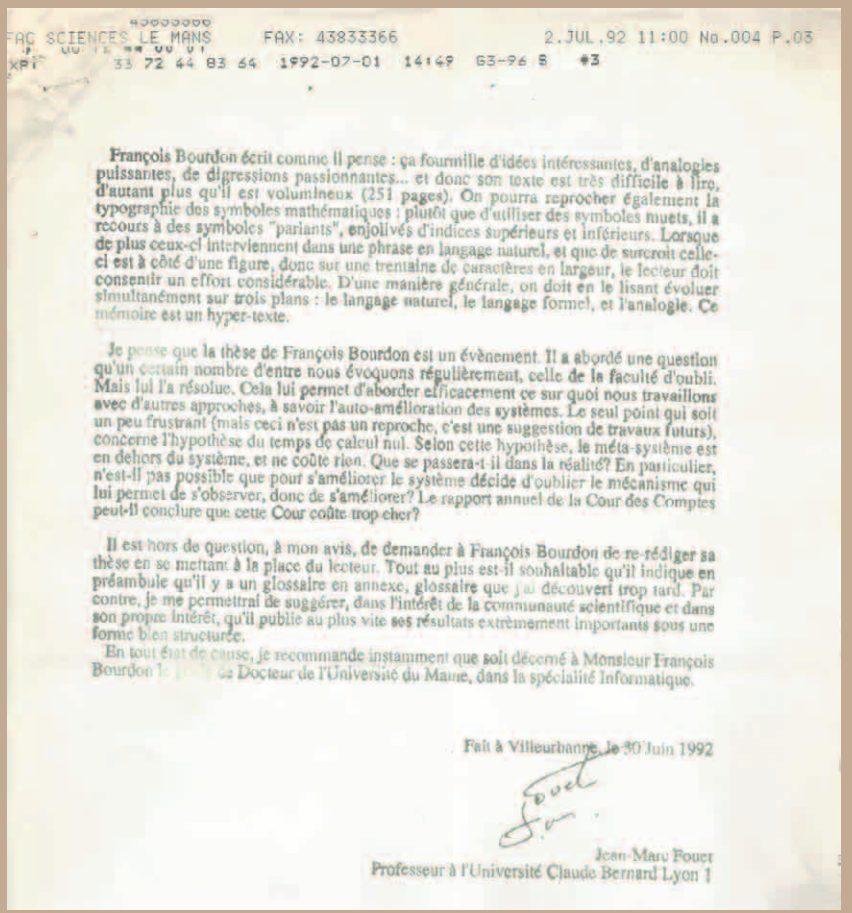
1991

Croquis de travail pendant les trois années de thèse.





1992



FAC SCIENCES LE MANS FAX: 43833366 2.JUL.92 11:00 No.004 P.03
33 72 44 83 64 1992-07-01 14:49 G3-96 S #3

François Bourdon écrit comme il pense : ça fourmille d'idées intéressantes, d'analogies puissantes, de digressions passionnantes... et donc son texte est très difficile à lire, d'autant plus qu'il est volumineux (251 pages). On pourra reprocher également la typographie des symboles mathématiques : plutôt que d'utiliser des symboles muets, il a recours à des symboles "parlants", enjolivés d'indices supérieurs et inférieurs. Lorsque de plus ceux-ci interviennent dans une phrase en langage naturel, et que de surcroît celle-ci est à côté d'une figure, donc sur une trentaine de caractères en largeur, le lecteur doit consentir un effort considérable. D'une manière générale, on doit en le lisant évoluer simultanément sur trois plans : le langage naturel, le langage formel, et l'analogie. Ce mémoire est un hyper-texte.

Je pense que la thèse de François Bourdon est un évènement. Il a abordé une question qu'un certain nombre d'entre nous évoquons régulièrement, celle de la faculté d'oubli. Mais lui l'a résolue. Cela lui permet d'aborder efficacement ce sur quoi nous travaillons avec d'autres approches, à savoir l'auto-amélioration des systèmes. Le seul point qui soit un peu frustrant (mais ceci n'est pas un reproche, c'est une suggestion de travaux futurs), concerne l'hypothèse du temps de calcul nul. Selon cette hypothèse, le méta-système est en dehors du système, et ne coûte rien. Que se passera-t-il dans la réalité? En particulier, n'est-il pas possible que pour s'améliorer le système décide d'oublier le mécanisme qui lui permet de s'observer, donc de s'améliorer? Le rapport annuel de la Cour des Comptes peut-il conclure que cette Cour coûte trop cher?

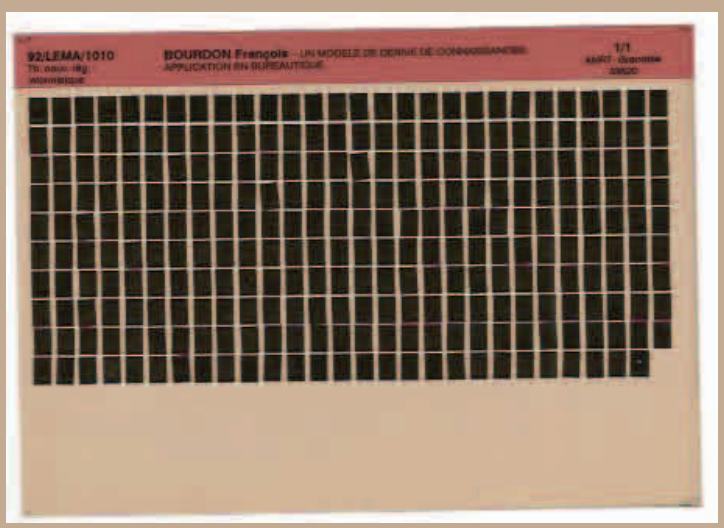
Il est hors de question, à mon avis, de demander à François Bourdon de re-rédiger sa thèse en se mettant à la place du lecteur. Tout au plus est-il souhaitable qu'il indique en préambule qu'il y a un glossaire en annexe, glossaire que j'ai découvert trop tard. Par contre, je me permettrai de suggérer, dans l'intérêt de la communauté scientifique et dans son propre intérêt, qu'il publie au plus vite ses résultats extrêmement importants sous une forme bien structurée.

En tout état de cause, je recommande instamment que soit décerné à Monsieur François Bourdon le [illegible] de Docteur de l'Université du Maine, dans la spécialité Informatique.

Fait à Villeurbanne, le 30 Juin 1992

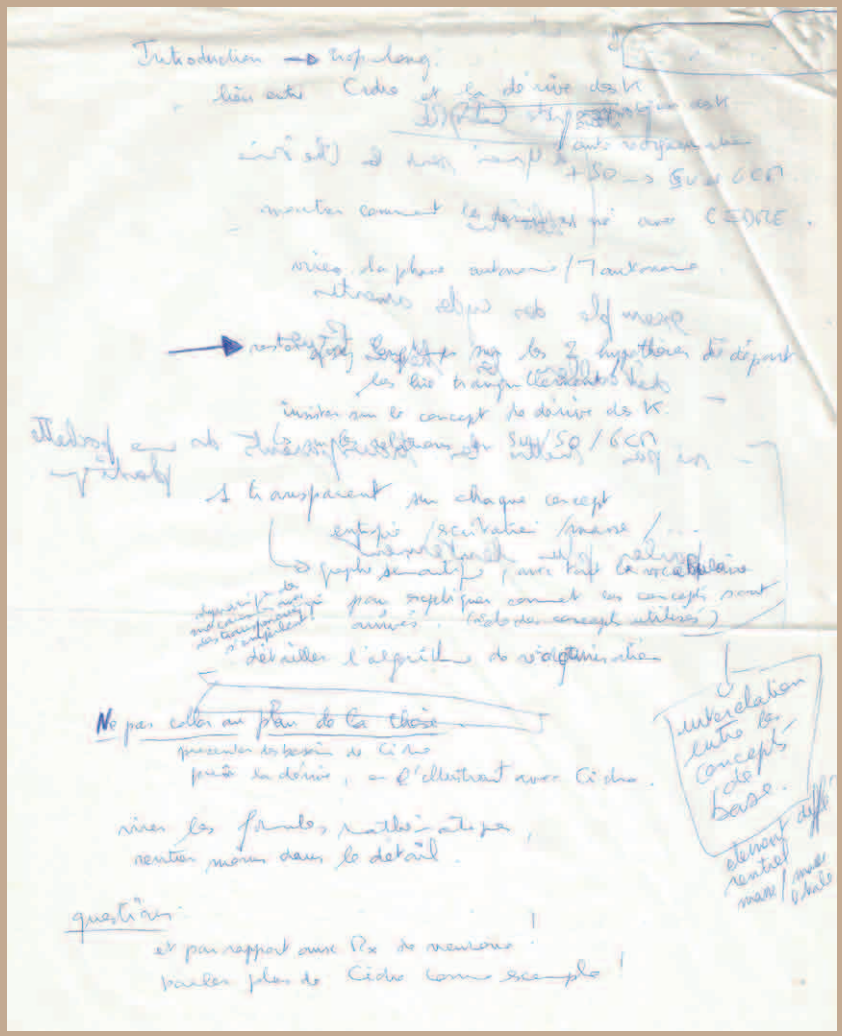
Jean-Marc Fouet
Professeur à l'Université Claude Bernard Lyon 1

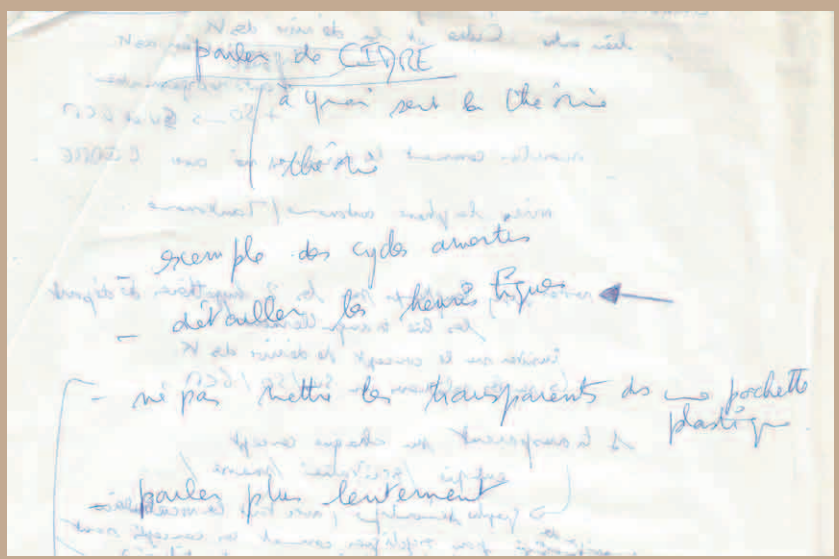
La fin du rapport sur ma thèse écrit par Jean-Marc Fouet. Pour ma petite histoire, je n'ai pratiquement rien publié sur ma thèse après la soutenance…
Jean-Marc Fouet avait soutenu sa thèse en 1987, intitulée «Utilisation de connaissances pour améliorer l'utilisation de connaissances: la machine Gosseyn», sous la direction de Jacques Pitrat, l'un des pionniers français de l'intelligence artificielle symbolique, qui s'intéressait aux systèmes à base de connaissances, aux démonstrateurs de théorèmes et aux métaconnaissances.



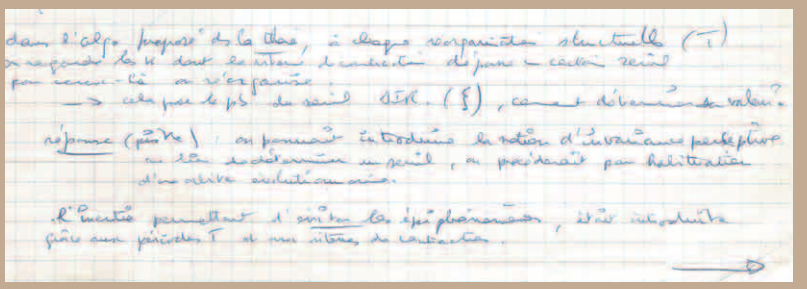
Ma thèse version microfiche.

Préparation de la soutenance de thèse… sur du papier bible !

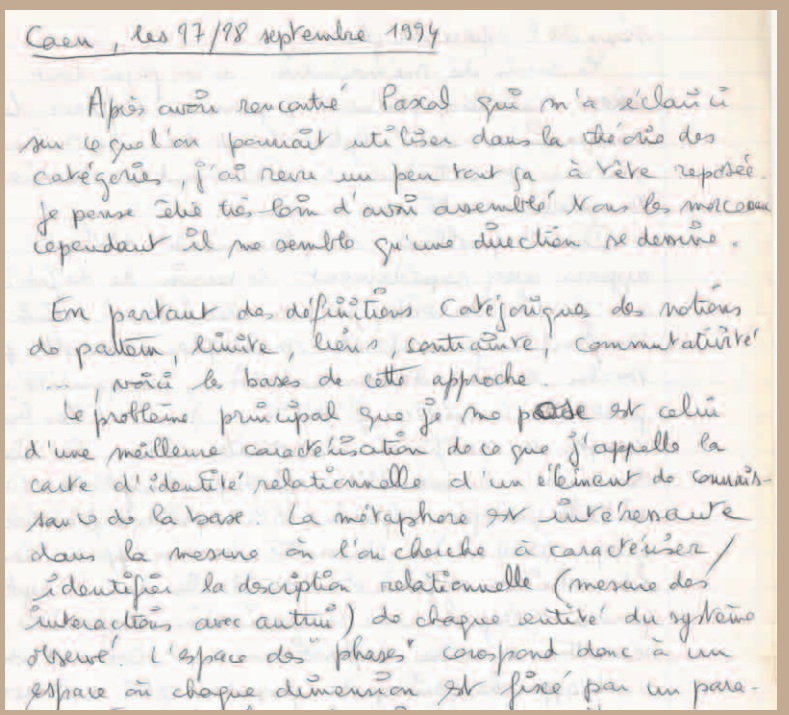




1994


dans l'algo proposé de la thèse, à chaque composition structurelle (T)
on regarde les K dont les intensités de contact dépassent le critère seuil
pour remonter là on s'organise.
→ cela pose le pb du seuil SIK. (§), comment déterminer sa valeur?
réponse (pour Ne) on pourrait introduire la notion d'invariance perceptive
en lien avec détermination du seuil, on procéderait par habituation
d'une activité exploratoire aléa.
l'heuristique permettant d'associer les épiphénomènes, devrait introduire
priorité aux paires T et aux intensités de contact.

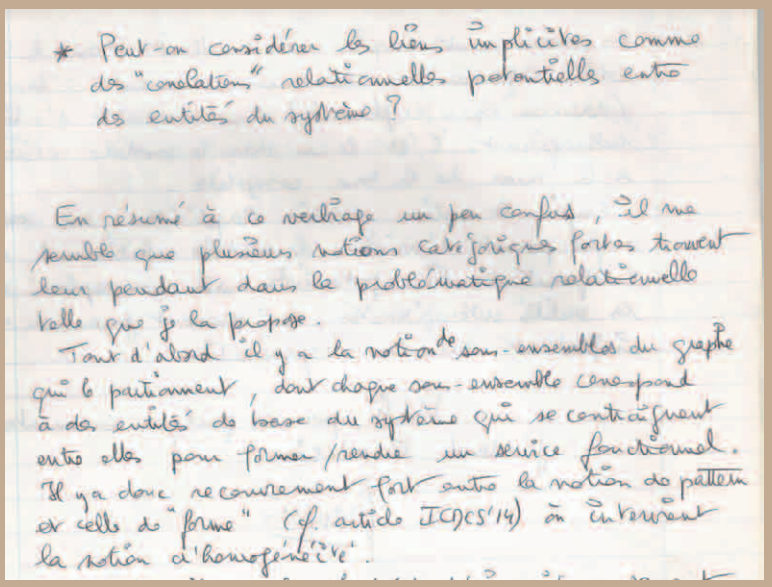
15 septembre 1994


Caen, les 97/98 septembre 1994

Après avoir rencontré Pascal qui m'a éclairé sur ce que l'on pourrait utiliser dans la théorie des catégories, j'ai revu un peu tout ça à tête reposée. Je pense être très loin d'avoir assimilé. Nous les mécanismes cependant il me semble qu'une direction se dessine.

En partant des définitions catégoriques des notions de pattern, limite, liens, contracture, commutativité ..., voici les bases de cette approche.
Le problème principal que je me pose est celui d'une meilleure caractérisation de ce que j'appelle la carte d'identité relationnelle d'un élément de connaissance de la base. La métaphore est intéressante dans la mesure où l'on cherche à caractériser / identifier la description relationnelle (mesure des interactions avec autrui) de chaque entité du système observé. L'espace des "phases" correspond donc à un espace où chaque dimension est fixé par un para-

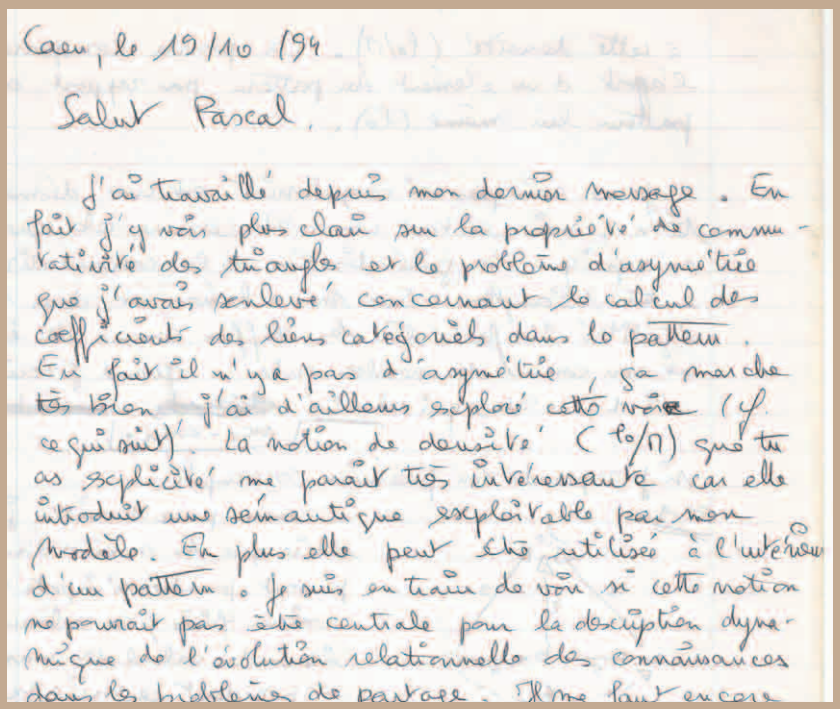
Extraits d'une correspondance avec Pascal Boldini, du laboratoire d'Intelligence Artificielle de l'ENST-Bretagne à Brest, à propos de la théorie mathématique des catégories appliquée au modèle relationnel, développé dans ma thèse.


* Peut-on considérer les liens implicites comme des "corrélations" relationnelles potentielles entre des entités du système ?

En résumé à ce verbiage un peu confus, il me semble que plusieurs notions catégoriques portes trouvent leur pendant dans la problématique relationnelle telle que je la propose.
Tout d'abord il y a la notion de sous-ensembles du graphe qui le partitionnent, dont chaque sous-ensemble correspond à des entités de base du système qui se contractent entre elles pour former / rendre un service fonctionnel. Il y a donc recouvrement fort entre la notion de pattern et celle de "forme" (cf. article ICDCS'14) en intervenant la notion d'homogénéité.

28 septembre 1994

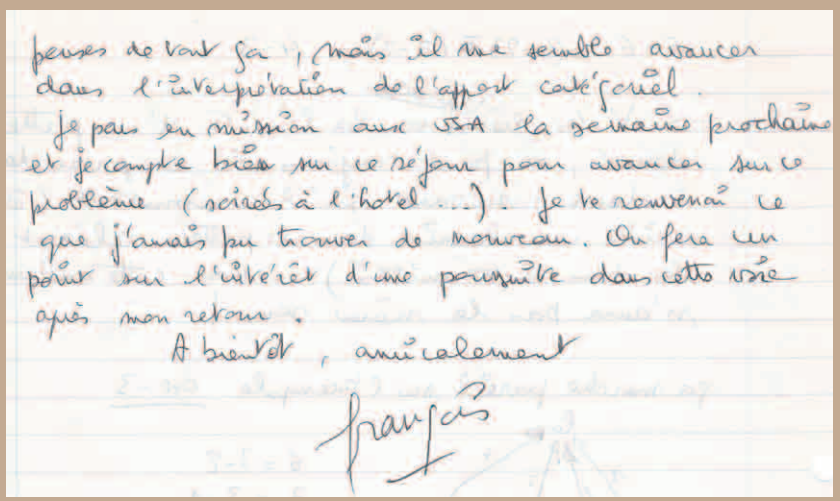
19 octobre 1994



Caen, le 19/10 94

Salut Pascal,

J'ai travaillé depuis mon dernier message. En fait j'y vois plus clair sur la propriété de commutativité des triangles et le problème d'asymétrie que j'avais soulevé concernant le calcul des coefficients des liens catégoriels dans le pattern. En fait il n'y a pas d'asymétrie, ça marche très bien. J'ai d'ailleurs exploré cette voie (cf. ce qui suit). La notion de densité (f/n) que tu as explicitée me paraît très intéressante car elle introduit une sémantique exploitable par mon modèle. En plus elle peut être utilisée à l'intérieur d'un pattern. Je suis en train de voir si cette notion ne pourrait pas être centrale pour la description dynamique de l'évolution relationnelle des connaissances dans les bibliothèques de partage. Il me faut encore

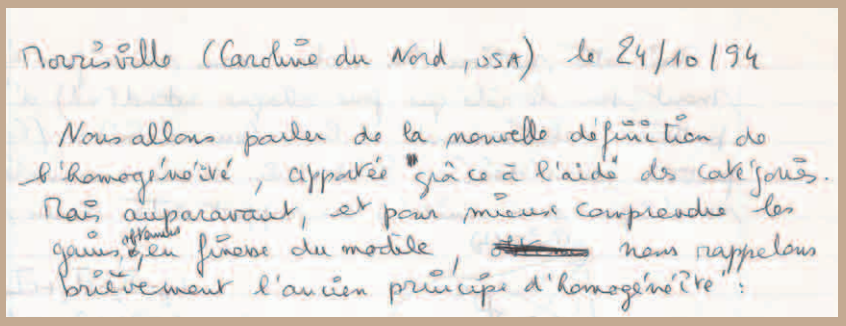


penser de tout ça, mais il me semble avancer dans l'interprétation de l'apport catégoriel.

Je pars en mission aux USA la semaine prochaine et je compte bien sur ce séjour pour avancer sur ce problème (soirées à l'hotel ...). Je te renverrai ce que j'aurais pu trouver de nouveau. On fera un point sur l'intérêt d'une poursuite dans cette voie après mon retour.

A bientôt, amicalement

François



Norrisville (Caroline du Nord, USA) le 24/10/94

Nous allons parler de la nouvelle définition de l'homogénéité, apportée grâce à l'aide des catégories. Mais auparavant, et pour mieux comprendre les gains, en finesse du modèle, ~~[illegible]~~ nous rappelons brièvement l'ancien principe d'homogénéité :

Il est de calculs catégoriels entre les limites (gerbes ...) qui doivent permettre de considérer qu'un pattern donné doit être inclus dans un autre.
Cette voie reste à explorer ...

Nouméville, 09/10/94

Avant d'aller plus en avant sur le calcul d'appréciation de pattern, il faut préciser le mécanisme général d'apprentissage auquel est soumis le système d'observation. Ceci explique l'initialisation des choses et donc l'apport de nouvelles connaissances dans le système et leur interprétation aux suivantes.
Nous venons de voir que le système est en perpétuelle recherche de point de rupture d'homogénéité relationnelle. On peut dire autrement qu'il est en phase d'apprentissage permanente.
Les observations de rupture d'homogénéité se font toujours sur une période (ΔT) dont la durée va dans un premier temps (1ère version du modèle) dépendre du contexte applicatif où l'on veut appliquer ce modèle.

Pascal (téléphone) le 14/11/94

Le problème catégoriel est que les valeurs attachées aux liens dans les patterns (flèches "d'en bas") doivent être absolues, càd fixes ∀ les patterns dans lesquels ces flèches se trouvent. Ce sont uniquement les flèches vers les limites qui ont des valeurs variables pour refléter les patterns.
pb → il faut donner un sens dans l'absolu aux flèches d'en bas.

objectif: éclaircir le point avant la fin du mois de novembre.

indépendamment des patterns [illegible] peut en venir !

Dijon, le 1/12/94

Je reprends la plume après une bonne semaine de vacances à l'âme catéjouelle ! J'ai bien réfléchi le problème soulevé par Pascal et concernant les fameux coefficients adhoc sur les liens dans les patterns. En effet en pensant à un calcul ~~qui~~ qui aillait au-delà que sur la commutativité, pour lequel on fait on donne un sens très fort aux flèches allant vers les feuilles (participation du nœud correspondant dans l'accroissement global de masse du pattern adhoc) et surtout on obtient des coefficients adhoc sur les flèches incluses dans les patterns, dont la sémantique n'est pas claire (différence entre les variations de masse dues aux nœuds reliés). Comble de désespoir, les coefficients changent suivant que l'on considère tel ou tel pattern reconnaissant le graphe, ce qui est contradictoire avec le fait que ce sont précisément les liens inclus dans les patterns qui doivent être identifiés à la partitionnement en patterns du graphe.

Mardi 6 décembre 1994

Salut Pascal,

Je reprends la plume pour tenter d'avancer un peu !

L'idée fondamentale de mon approche est de considérer les nœuds de mon graphe relationnel comme étant des entités "pseudo-autonomes" dans la dynamique d'évolution de leur masse relationnelle. Cela signifie que l'influence en terme d'évolution de masse relationnelle des nœuds entre eux ne se fait qu'à un seul niveau (celui de la visibilité immédiate), c-à-d. sans propagation.

"a voit b", qui voit c, qui voit d
la masse de "a" dépend uniquement de la partie constante de la masse de "b"

Je m'arrête là pour aujourd'hui et j'attends vos réactions. Je n'ai pas développé toutes les notions induites par la notion de dérivée telle que je l'avais présentée auparavant et qui restent tout à fait valides dans ce contexte. On a la dérivée d'un nœud vis-à-vis de chaque pattern dans lequel il est et on a bien sûr vis-à-vis de sa position dans le réseau ; on a également la dérivée d'un pattern vis-à-vis de la base ...
Il faudrait aussi développer tout ce qui concerne le calcul de perte ...
Il me semble cette fois que la description catéjouelle tient la route et s'avère riche et prometteuse dans ce contexte.
A Bientôt.

François

1995

Argentière, le 11/4/95

Je poursuis l'idée que j'ai présentée à Rochebrune sur la détection de formes stables (pattern) à partir de points de vue locaux (référentiels locaux d'accessibilité). Cette idée est d'utiliser l'invariance perceptive pour que chaque nd du RK se construise une représentation des effets qu'il procure au delà de sa visibilité (nds qu'il sait liés). En effet le problème des pattern ou formes collectives stables, c'est qu'elles sont par définition, multi-référentiels locaux.

Réflexions sur la notion de forme stable en représentation des connaissances.

Argentière le 15/4/95

Nous avons été voir l'exposition d'Ejon Schiele à Martigny en Suisse, c'est assez impressionnant et dur à la fois. J'ai repris le bouquin de Parochian sur la cosmogénèse de l'information. Il est vraiment dans la ligné des discussions que j'ai eu avec Roger Cauzien à Rochebrune. Il faudrait que Roger m'explique certaines choses pour que je comprenne un peu mieux ce livre.

Depuis que je suis à Argentière je tourne autour de mon grande unification, je sens qu'il y a vraiment quelque chose à faire, mais que cela risque de prendre pas mal de temps!

Pour cela je reprend l'exploration vers azimuts de cette pré-unification informationnelle (cf. mon article de déc 94)

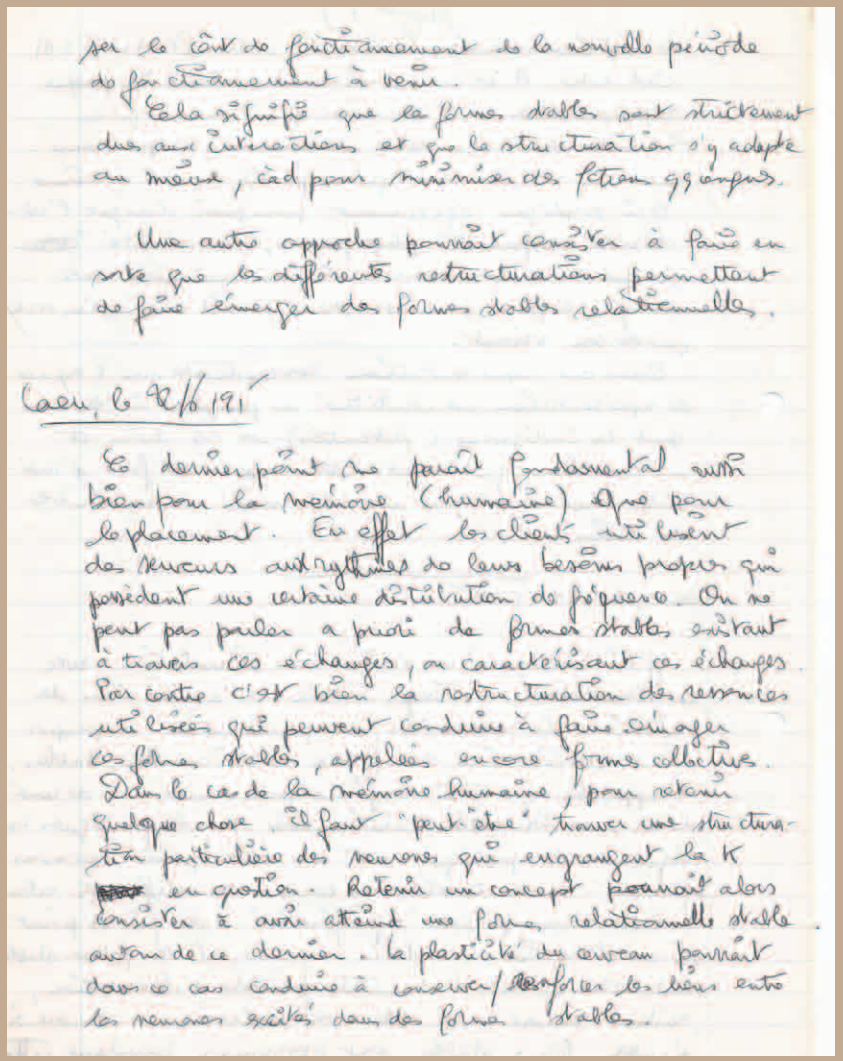
Je reviens sur mes plans saturants, espace réel et espace relationnel décrits à Rochebrune et reprends la métaphore de l'individu Romain (un peu d'anthropomorphisme ne peut pas faire de mal, surtout ici!).

Caen, le 11/5/95

J'ai commencé à écrire une simulation avec le simulateur de Roger. Cela m'amène dans des réflexions intéressantes, en particulier à propos de l'utilisation des formes relationnelles stables. L'approche que j'ai toujours poursuivie est de considérer une observation relationnelle sur les pratiques en réseau. Ces pratiques ont un côté précis que nous fournit la structuration au cours des différents actes. Au bout d'une période "significative", on peut le point sur ces observations, càd on regarde si les formes stables saillantes au début de cette période d'observation, subsistent encore, si elles ont évoluées ou encore si d'autres formes stables sont apparues pendant cette période. Le travail consiste alors à ajuster la structuration des acteurs au cause pour minimi- (33)

Début des simulations avec oRis.

1996



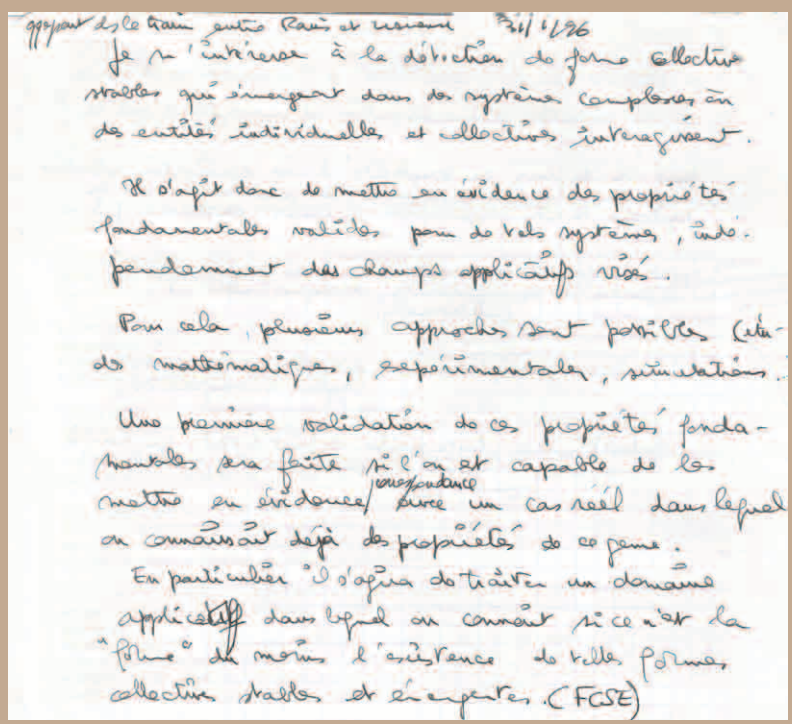
ser le cout de fonctionnement de la nouvelle période de fonctionnement à venir.

Cela signifie que les formes stables sont structurées par les interactions et que la structuration s'y adapte au mieux, càd pour minimiser des fonctions goulues.

Une autre approche pourrait consister à faire en sorte que les différentes restructurations permettent de faire émerger des formes stables relativement.

Caen, le 12/6/91

Ce dernier point me paraît fondamental aussi bien pour la mémoire (humaine) que pour le placement. En effet les clients utilisent des acteurs aux rythmes de leurs besoins propres qui possèdent une certaine distribution de fréquence. On ne peut pas parler a priori de formes stables existant à travers ces échanges, ou caractérisant ces échanges. Par contre c'est bien la restructuration des ressources utilisées qui peuvent conduire à faire émerger ces formes stables, appelées encore formes collectives. Dans le cas de la mémoire humaine, pour retenir quelque chose, il faut "peut-être" traiter une structuration particulière des neurones qui engramment la K en question. Retenir un concept pourrait alors consister à avoir atteint une forme relationnelle stable autour de ce dernier. La plasticité du cerveau pourrait dans ce cas conduire à conserver/renforcer les liens entre les neurones excités dans des formes stables.



rapport de la thèse entre Paris et Caen 31/1/96

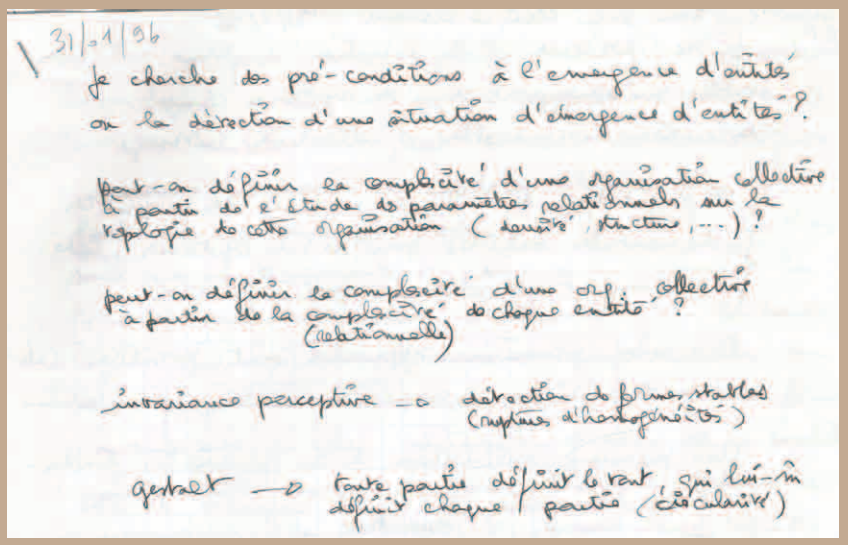
Je m'intéresse à la détection de formes collectives stables qui émergent dans des systèmes complexes où des entités individuelles et collectives interagissent.

Il s'agit donc de mettre en évidence des propriétés fondamentales valides pour de tels systèmes, indépendamment des champs applicatifs visés.

Pour cela plusieurs approches sont possibles (étude mathématique, expérimentale, simulation...).

Une première validation de ces propriétés fondamentales sera faite si l'on est capable de les mettre en évidence/correspondance avec un cas réel dans lequel on connaissait déjà des propriétés de ce genre.

En particulier il s'agira de traiter un domaine applicatif dans lequel on connaît si ce n'est la "forme" du moins l'existence de telles formes collectives stables et émergentes. (FCSE)



31/01/96

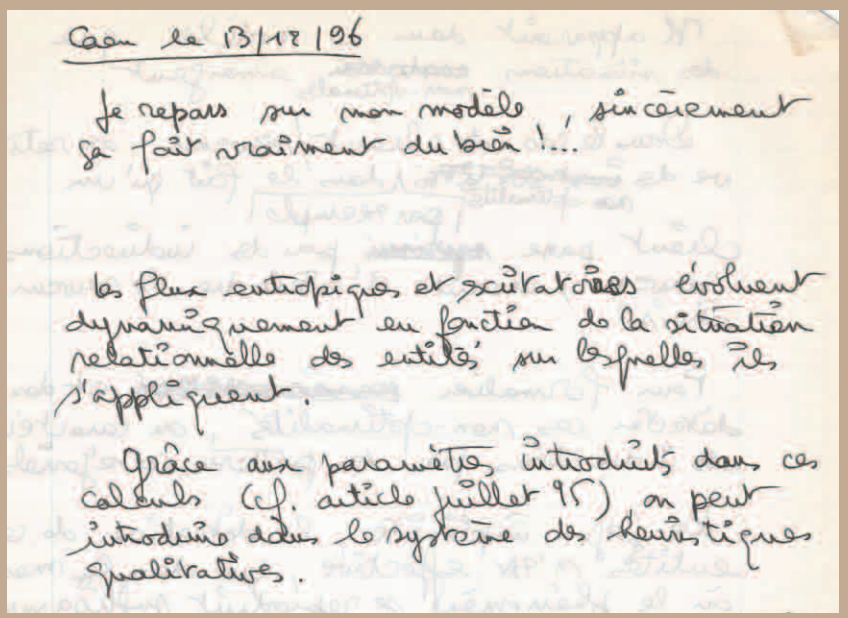
Je cherche des pré-conditions à l'émergence d'entités ou la direction d'une situation d'émergence d'entités ?

peut-on définir la complexité d'une organisation collective à partir de l'étude de paramètres relationnels sur la topologie de cette organisation (densité, structure, ...) ?

peut-on définir la complexité d'une org. collective à partir de la complexité (relationnelle) de chaque entité ?

invariance perceptive → direction de formes stables (ruptures d'homogénéités)

gestalt → toute partie définit le tout, qui lui-même définit chaque partie (circularité)



Caen le 13/12/96

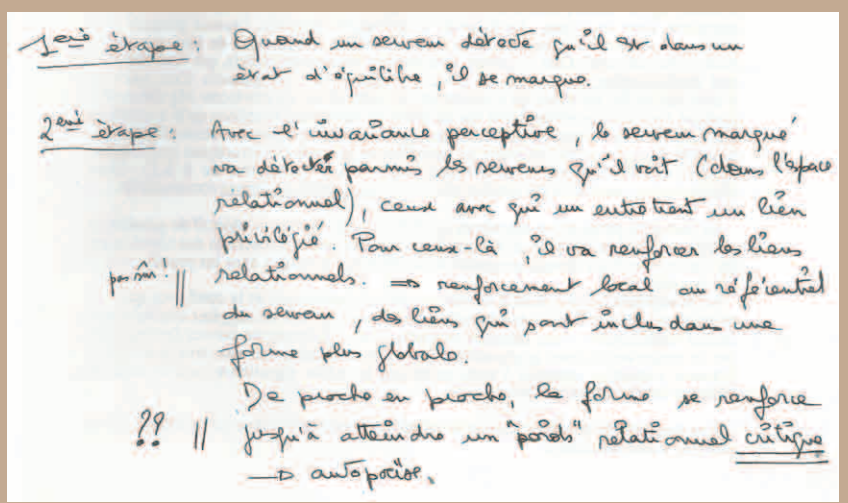
Je repars sur mon modèle, sincèrement ça fait vraiment du bien !...

les flux entropiques et accumulations évoluent dynamiquement en fonction de la situation relationnelle des entités sur lesquelles ils s'appliquent.

Grâce aux paramètres introduits dans ces calculs (cf. article juillet 95) on peut introduire dans le système des heuristiques qualitatives.

Début des réflexions sur les heuristiques qualitatives.

1997



1ère étape : Quand un neurone détecte qu'il est dans un état d'équilibre, il se marque.

2ème étape : Avec l'invariance perceptive, le neurone marqué va détecter parmi les neurones qu'il voit (dans l'espace relationnel), ceux avec qui il entretient un lien privilégié. Pour ceux-là, il va renforcer les liens relationnels. ⇒ renforcement local au référentiel du neurone, des liens qui sont inclus dans une forme plus globale.

pas fini !

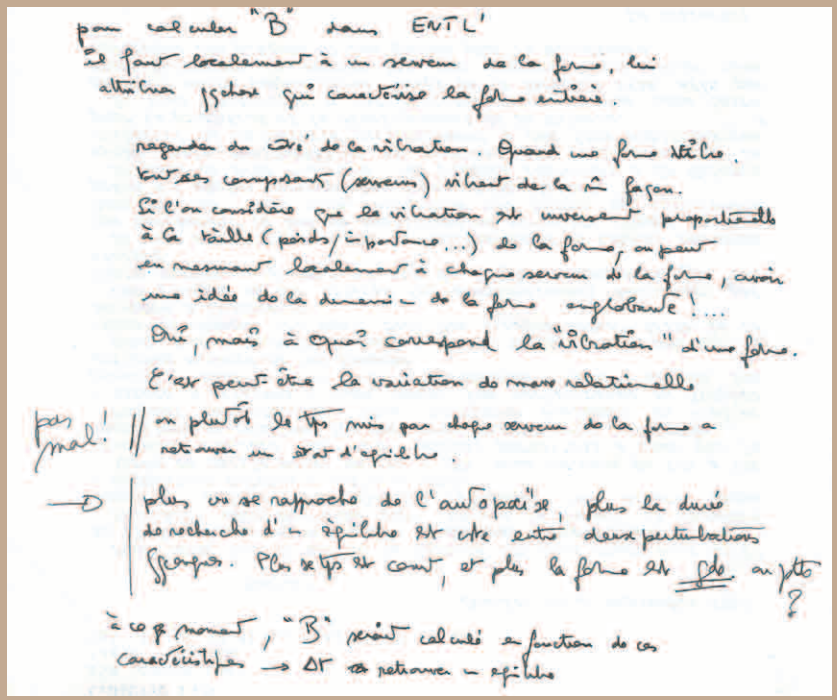
?? De proche en proche, la forme se renforce jusqu'à atteindre un "point" relationnel critique → autopoïèse.

Apparition du concept d'autopoïèse.

1998

États d'âmes existentiels.

Je confirme a posteriori…



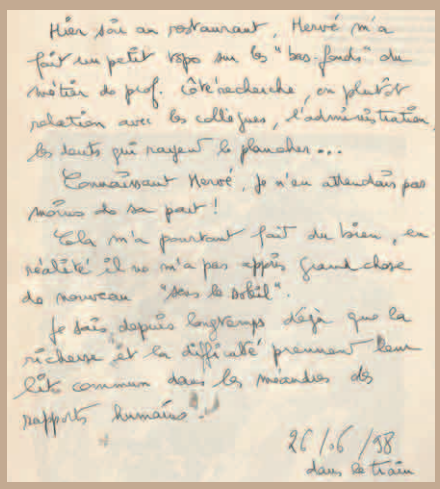
pour calculer "B" dans ENTL'
il faut localement à un serveur de la ferme, lui attribuer ce que caractérise la ferme entière.

regardons du côté de la vibration. Quand une ferme vibre, tous ses composants (serveurs) vibrent de la même façon.
Si l'on considère que la vibration est inversement proportionnelle à la taille (poids/inertie...) de la ferme, on peut en mesurant localement à chaque serveur de la ferme, avoir une idée de la dynamique de la ferme englobante !...
Oui, mais à quoi correspond la "vibration" d'une ferme.
C'est peut-être la variation de mesures relationnelles

pas mal! // ou plutôt le temps mis par chaque serveur de la ferme à retrouver un état d'équilibre.

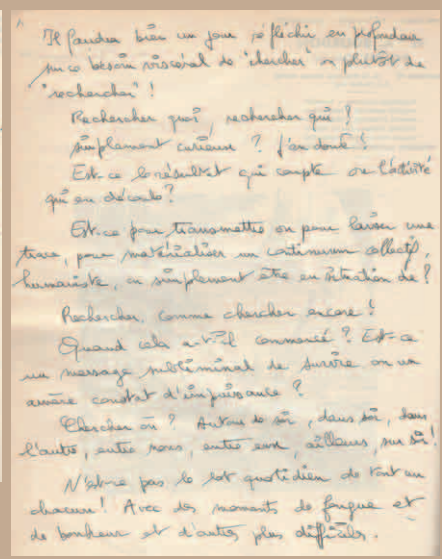
→ plus on se rapproche de l'autopoïèse, plus la durée de recherche d'un équilibre est entre deux perturbations longues. Plus ce temps est court, et plus la ferme est fde ou pté ?

à ce moment, "B" serait calculé en fonction de ces caractéristiques → ΔT de retour à équilibre

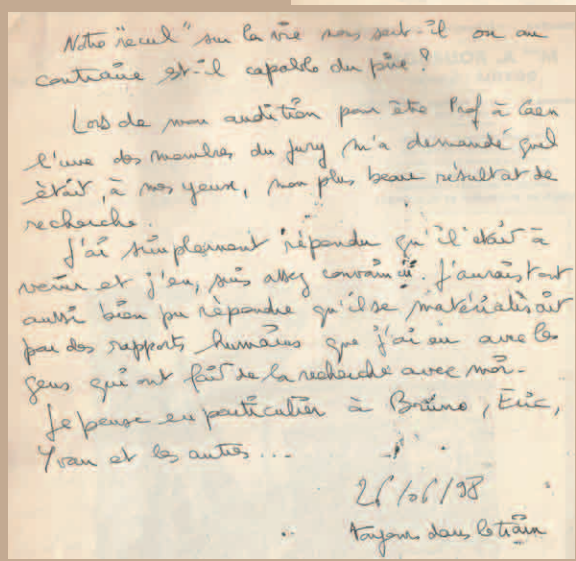


Hier soir au restaurant, Hervé m'a fait un petit topo sur les "bas-fonds" du milieu du prof. Côté recherche, ou plutôt relation avec les collègues, l'administration, les sous qui rayent le plancher...
Connaissant Hervé, je n'en attendais pas moins de sa part !
Cela m'a pourtant fait du bien, en réalité il ne m'a pas appris grand chose de nouveau "dans le fond".
Je sais depuis longtemps déjà que la richesse et la difficulté prennent leur lit commun dans les méandres des rapports humains.

26/06/98
dans le train



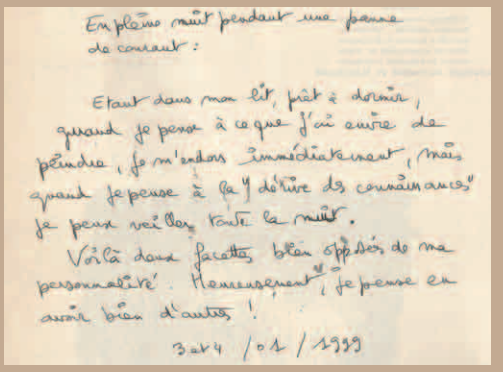
Il faudra bien un jour se pencher en profondeur sur ce besoin viscéral de "chercher" ou plutôt de "rechercher" !
Rechercher quoi, rechercher qui ?
Simplement curieux ? J'en doute !
Est-ce le résultat qui compte ou l'activité qu'on déploie ?
Est-ce pour transmettre ou pour laisser une trace, pour matérialiser un continuum collectif, humaniste, ou simplement être en relation ?
Rechercher, comme chercher encore !
Quand cela a-t-il commencé ? Est-ce un message subliminal de survie ou un ancien contrat d'inconscient ?
Chercher où ? Autour de soi, dans soi, dans l'autre, entre nous, entre eux, ailleurs, nulle part !
N'est-ce pas le sort quotidien de tout un chacun ! Avec des moments de fugue et de bonheur et d'autres plus difficiles.



Notre "recul" sur la vie nous sert-il ou au contraire est-il capable du pire ?

Lors de mon audition pour être Prof à Caen l'une des membres du jury m'a demandé quel était, à mes yeux, mon plus beau résultat de recherche.
J'ai simplement répondu qu'il était à venir et j'en, sus alors convaincu. J'aurais tout aussi bien pu répondre qu'il se matérialisait par des rapports humains que j'ai eu avec les gens qui ont fait de la recherche avec moi.
Je pense en particulier à Bruno, Eric, Yvan et les autres...

26/06/98
Toujours dans le train


En pleine nuit pendant une panne de courant :

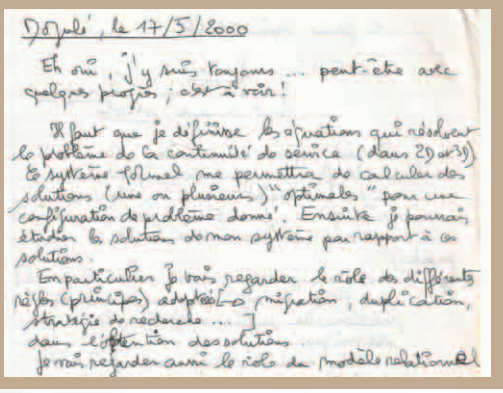
Etant dans mon lit, prêt à dormir, quand je pense à ce que j'ai envie de prendre, je m'endors immédiatement, mais quand je pense à la "désirée des connaissances" je peux veiller toute la nuit.

Voilà dans facettes bien opposées de ma personnalité. Heureusement, je pense en avoir bien d'autres !

3 et 4 / 01 / 1999

1999

Soit j'ai eu cette pensée «around midnight», soit elle s'est étirée dans la nuit…


Djuli, le 17/5/2000

Eh oui, j'y suis toujours ... peut-être avec quelques progrès ; à voir !

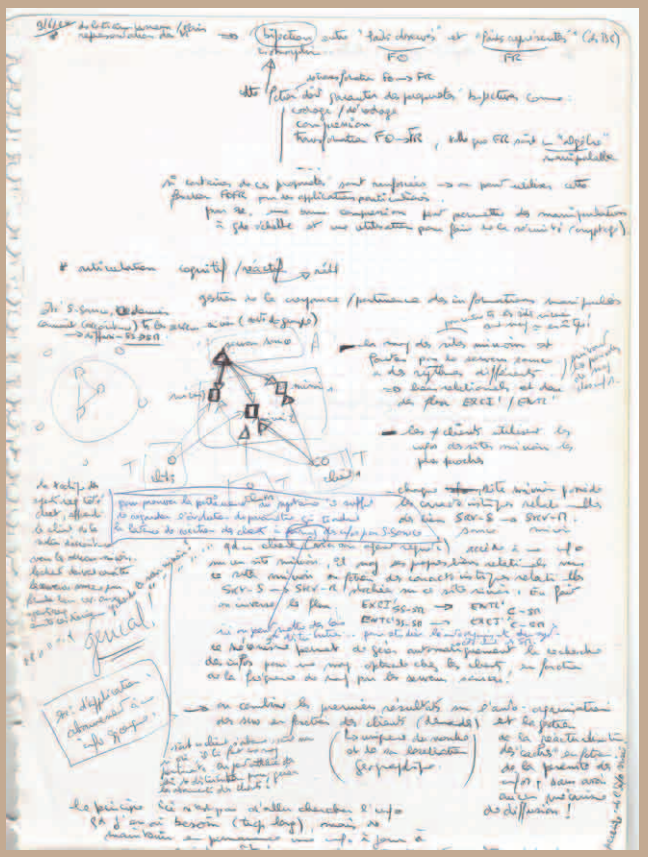
Il faut que je définisse les opérations qui résolvent le problème de la continuité de service (dans 2) et 3)). Le système formel me permettra de calculer des solutions (une ou plusieurs) "optimales" pour une configuration de problème donné. Ensuite je pourrai étudier la solution de mon système par rapport à ces solutions.

En particulier je vais regarder le rôle des différents règles (principes) adoptés : migration, duplication, stratégie de recherche ... dans l'optimisation des solutions.

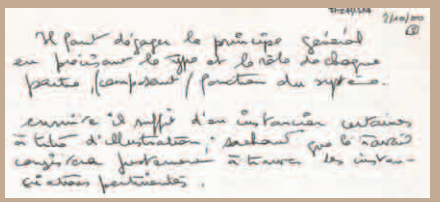
Je vais regarder aussi le rôle du modèle relationnel

2000

J'y crois encore…



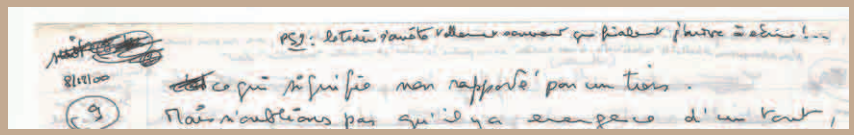
Un peu d'emballement, peut-être…
J'avoue que la recherche vous transbahute sans cesse entre euphorie et découragement, en passant par des phases plus calmes, et heureusement !

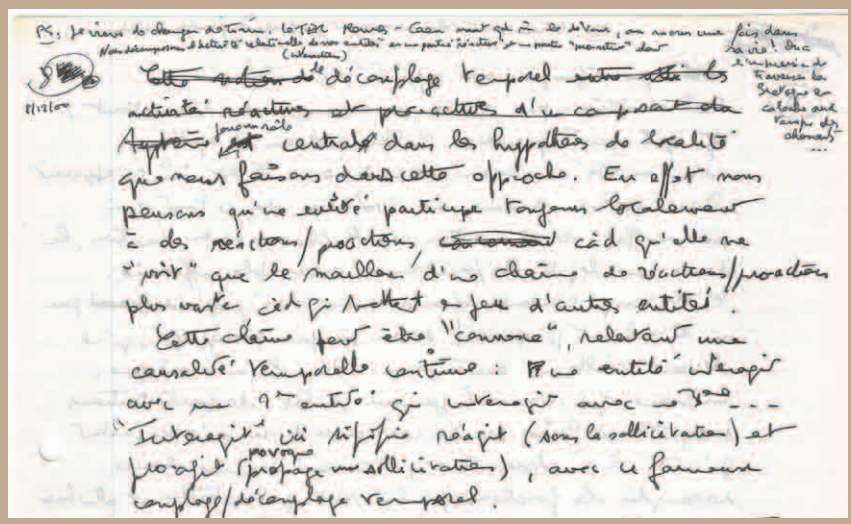

Il faut dégager le principe général en précisant le type et le rôle de chaque partie (composant / fonction du système).

Ensuite il suffit d'en instancier certains à titre d'illustration, sachant que le travail consistera finalement à trouver les instanciations pertinentes.

Cela se voit par l'emploi de mots un peu fort, comme ci-contre «dégager».

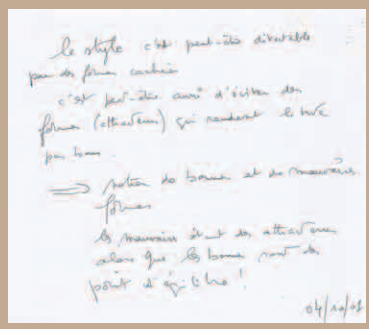
Le train aura été pour moi un haut lieu de divagation et de liberté intellectuelle !



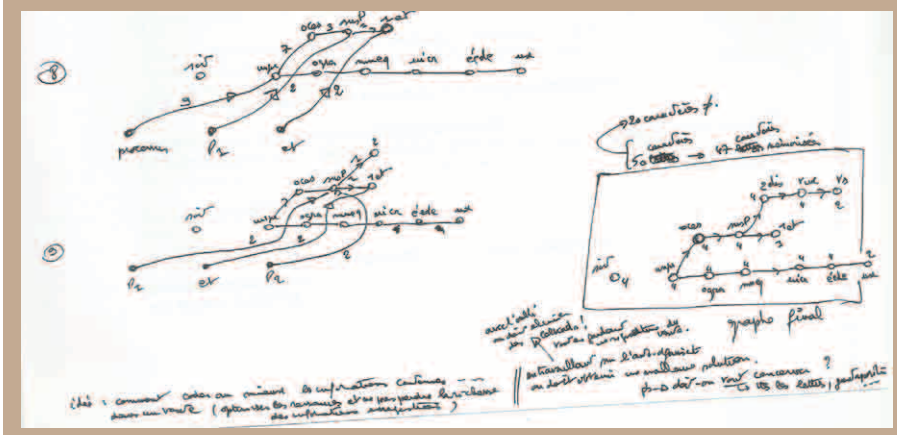


2001

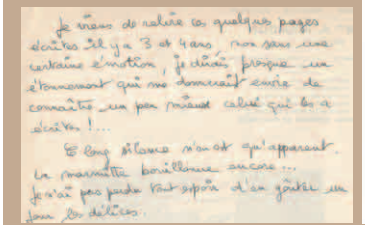
C'était à une période où je voulais étudier le style en littérature, ou plus exactement ce qu'il reste d'un livre au-delà du sens porté par le langage pour décrire un narratif. Je fais référence à ma rencontre avec les écrits de Leo Perutz.

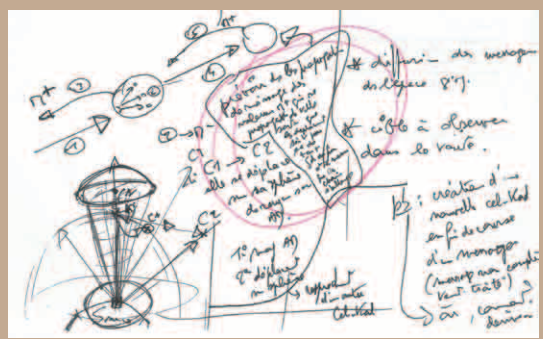


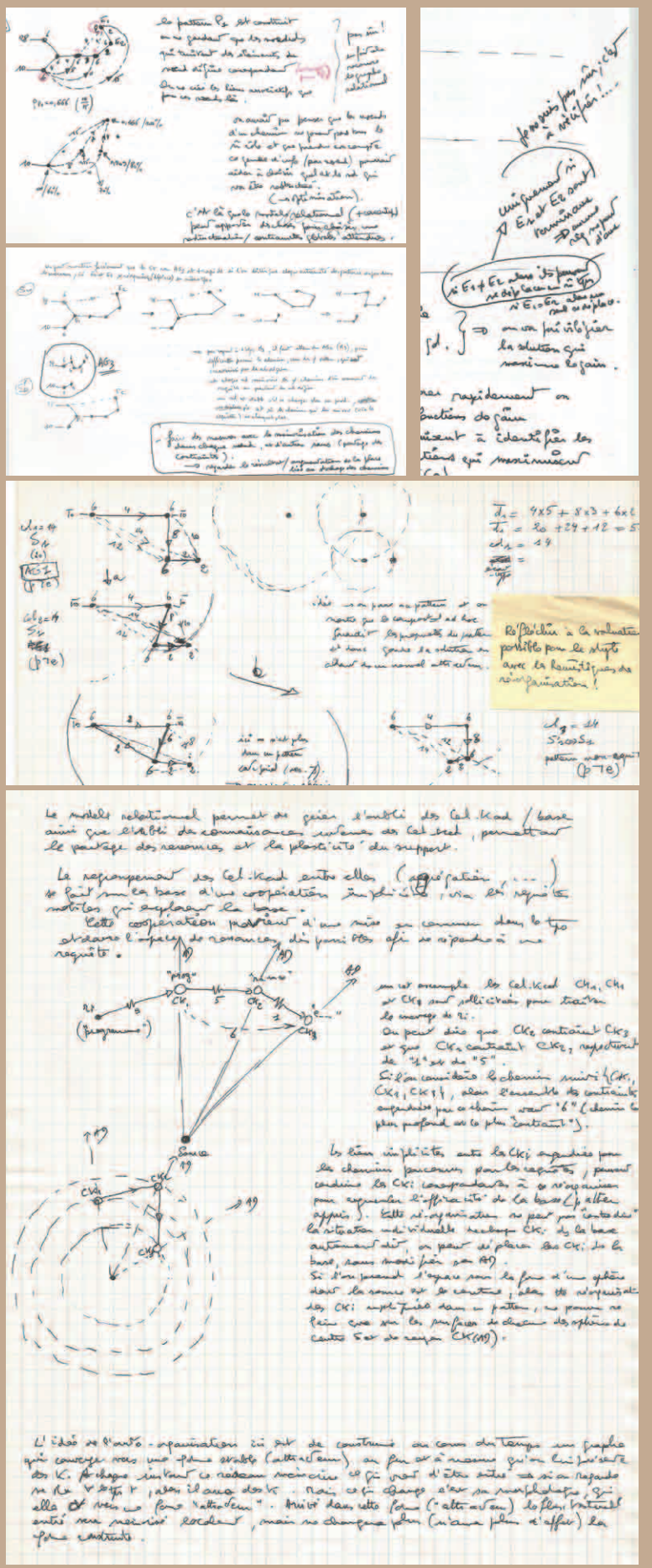
2002



Mon travail de recherche aura été multi-pistes, mais mono-directionnel. Ici je fais référence à l'une de ces nombreuses pistes.



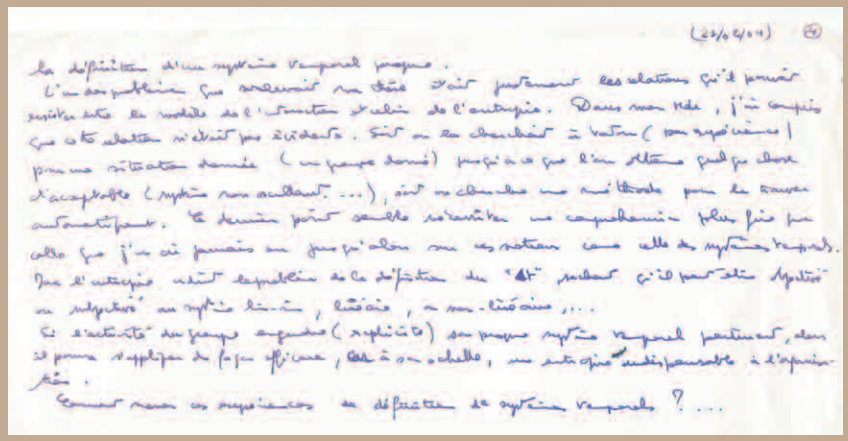


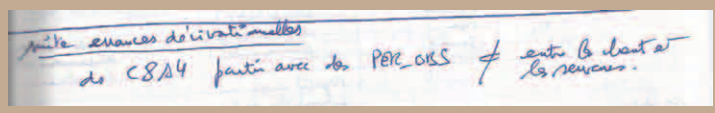


2003

J'ai vraiment voulu croire en l'approche catégorielle en mathématique, en particulier pour m'aider à travailler et définir la notion de forme stable dans la dynamique de construction et d'évolution des connaissances…
Cela restera pour moi comme une question ouverte, à l'image de la représentation des connaissances, via la dérive des connaissances !

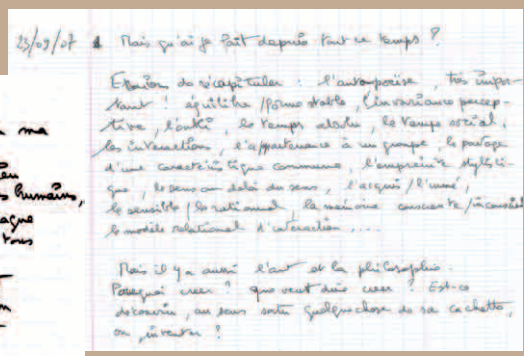
2004

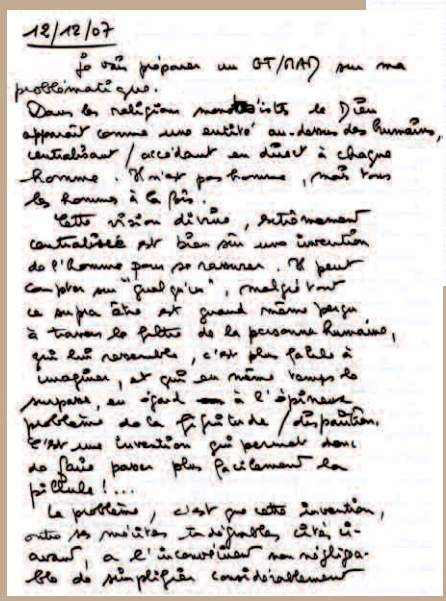




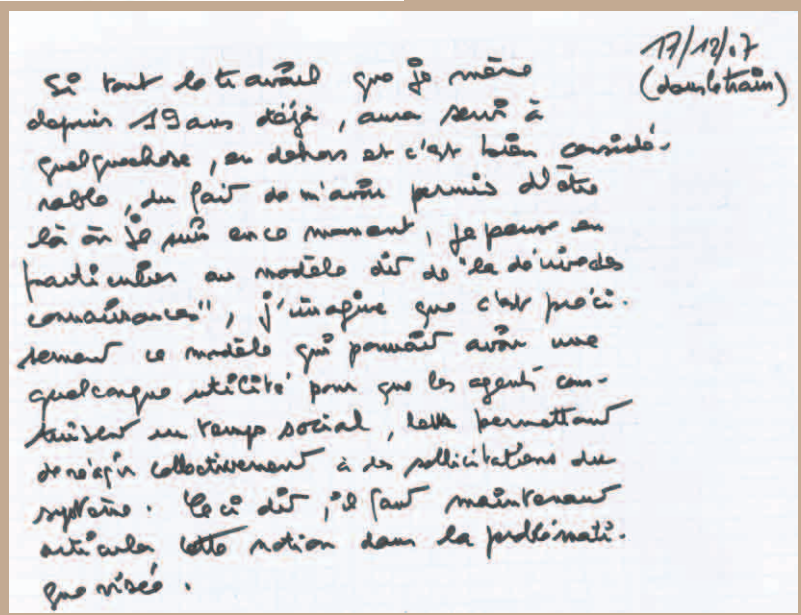
2007

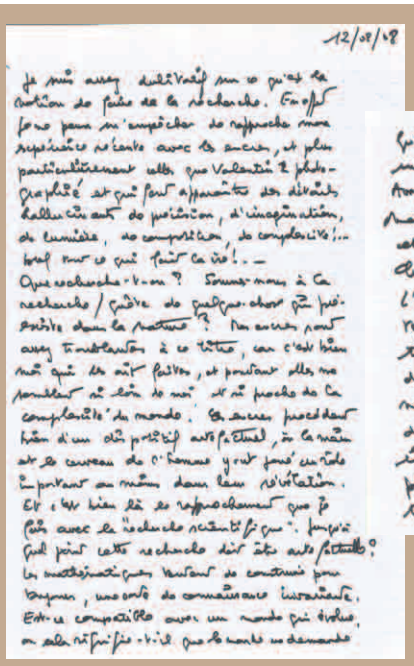
Et cette question lancinante qui me poursuit toujours et toujours, sur la création.
Pouvoir se la poser témoigne sans doute d'une preuve d'existence…





Le doute toujours…
le train aussi !

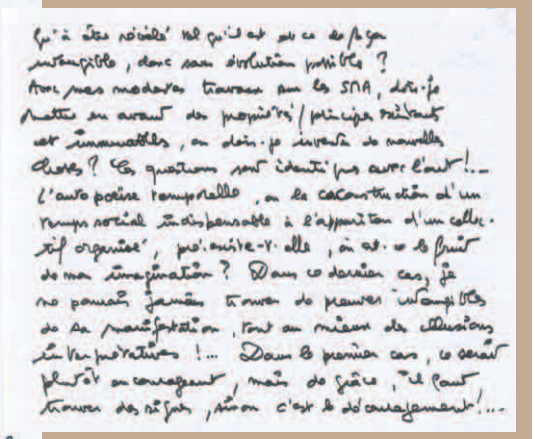




12/02/08

Je suis assez déstabilisé par ce qu'est la notion de fuite de la recherche. Ce n'est pas pour m'empêcher de rapprocher mon expérience récente avec les arcs, et plus particulièrement celle que Valentin a photographié et qui font apparaître des détails d'une extrême précision, d'imagination, de lumière, de complexité, de complexité !... bref tout ce qui fait la vie ! —

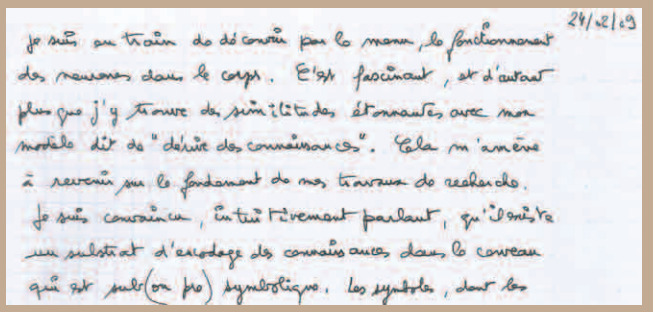
Que recherche-t-on ? Sommes-nous à la recherche / prise de quelque chose qui préexiste dans la nature ? Nos cerveaux sont avec tâtonnements à ce titre, car c'est bien eux qui les ont faites, et pourtant elles ne semblent si loin de nous et si proche de la complexité du monde. Ce cerveau procédant bien d'un développement extrêmement, à la main et au cerveau de l'homme y ont joué un rôle important au même dans leur révélation. Et c'est bien là le rapprochement que je fais avec la "recherche scientifique". Jusqu'à quel point cette recherche doit être subjective ? Les mathématiques seraient de construction pure, une sorte de connaissance innée ? Est-il compatible avec un monde qui évolue, on alors n'est-il que le monde se demande



qu'à être révélé tel qu'il est ou ce de façon intangible, donc sans évolution possible ?
Avec mes modestes travaux sur les SNA, dois-je mettre en avant des propriétés / principes existants et immuables, ou dois-je inventer de nouvelles choses ? Ces questions sont identiques avec d'autres !... L'auto poïèse temporelle, ou la construction d'un temps sociétal indispensable à l'apparition d'un collectif organisé, préexiste-t-elle, ou est-ce le fruit de mon imagination ? Dans ce dernier cas, je ne pourrais jamais trouver de preuves intangibles de sa manifestation, tout au mieux des illusions séductrices !... Dans le premier cas, ce serait plutôt encourageant, mais de grâce, il faut trouver des règles, sinon c'est à décourager !...

2008

Et toujours ce parallèle fécond entre la pratique de l'encre et le travail scientifique.

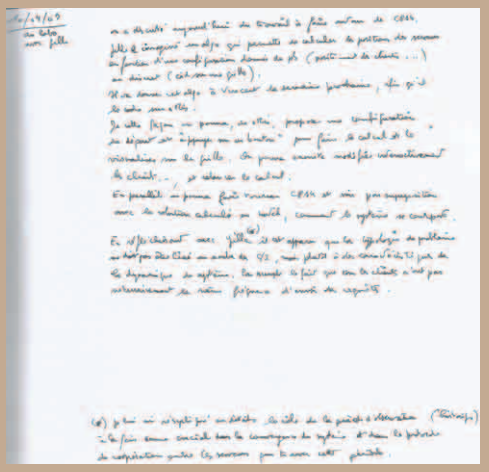


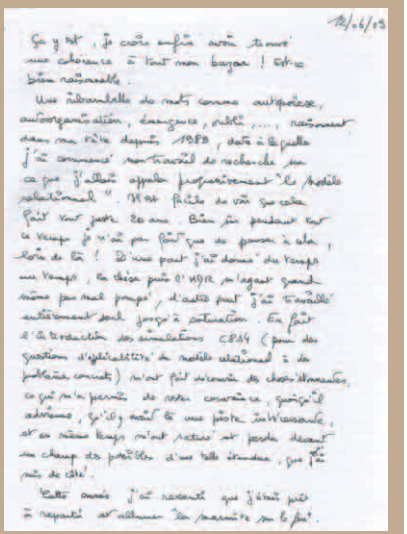
24/02/09

Je suis en train de découvrir par le menu le fonctionnement des neurones dans le corps. C'est fascinant, et d'autant plus que j'y trouve des similitudes étonnantes avec mon modèle dit de "déduction des connaissances". Cela m'amène à revenir sur le fondement de mes travaux de recherche. Je suis convaincu, intuitivement parlant, qu'il existe un substrat d'encodage des connaissances dans le cerveau qui est soit (ou pas) symbolique. Les symboles, dont les

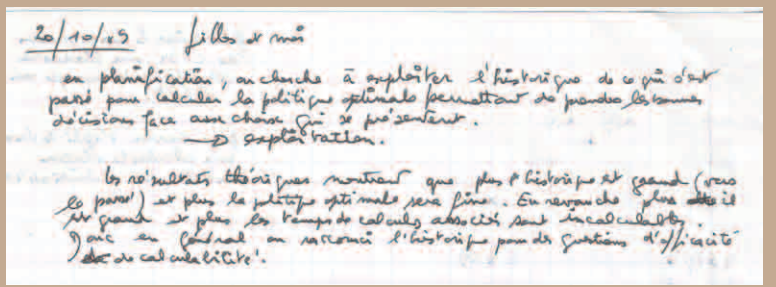
2009

Je me suis rapproché de la mécanique neuronale du cerveau humain et de son volet représentation symbolique.





Le recours aux simulations, entre émerveillement, étourdissement et vertige.



20/10/09 fille et moi

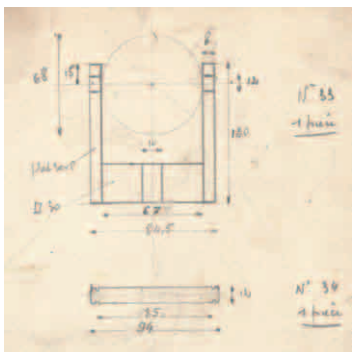
en planification, on cherche à exploiter l'historique de ce qui s'est passé pour calculer la politique optimale permettant de prendre les bonnes décisions face aux choses qui se présentent.
→ exploitation.

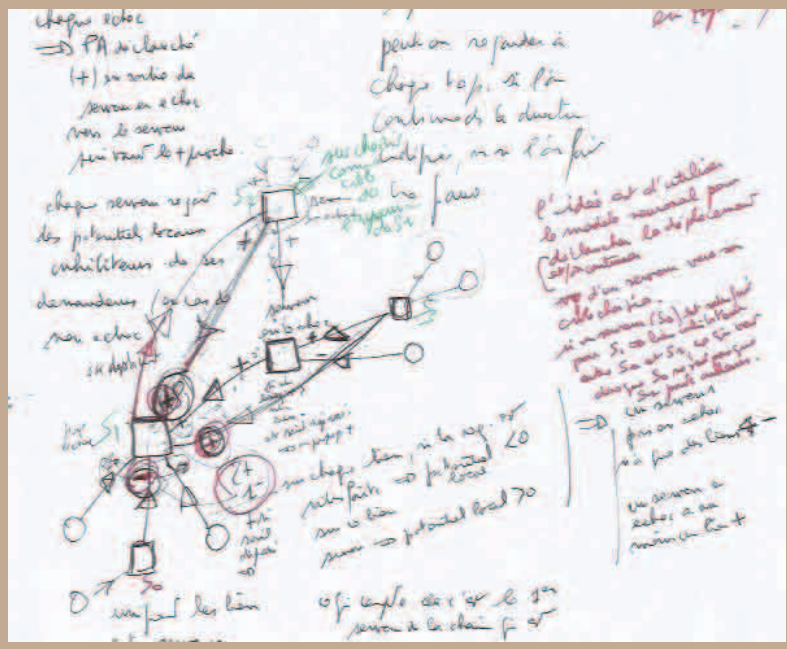
les résultats théoriques montrent que plus l'historique est grand (vers le passé) et plus la politique optimale sera fine. En revanche plus elle est grande et plus les temps de calculs associés sont incalculables.
Donc en général on raccourci l'historique pour des questions d'efficacité et de calculabilité.

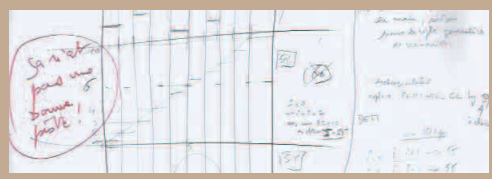
Premières réflexions sur ce que j'ai appelé l'empan de réactivité. C.-à-d. la fenêtre temporelle passée, sur laquelle on s'appuie pour orienter le futur immédiat.

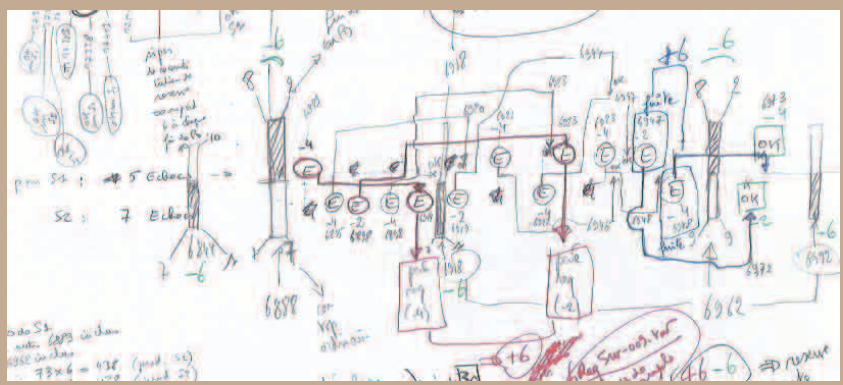
Travail exploratoire sur le rôle des échecs (entrants et sortants) dans la dynamique du système de résolution (recherche d'équilibre, coopération…).

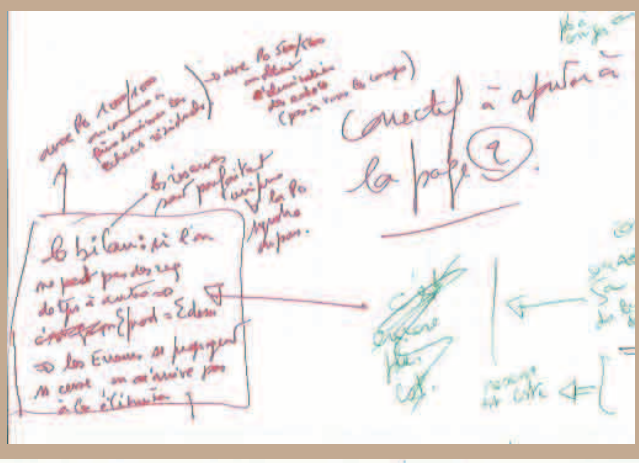
J'en profite pour souligner mon choix volontaire d'avoir présenté, dans ce document, une très grande partie des dessins sous leur forme originelle, c.-à-d. faits à la main.
Le crayon (générique) et le papier restent pour moi des outils extrêmement pertinents, au service de la pensée.
J'ajoute que c'est aussi un clin d'œil, voire un hommage, à mon père qui n'aura pas eu la chance de connaître l'informatique, et qui fit de très nombreux dessins à la main, comme celui ci-après, pour aboutir à des brevets déposés.

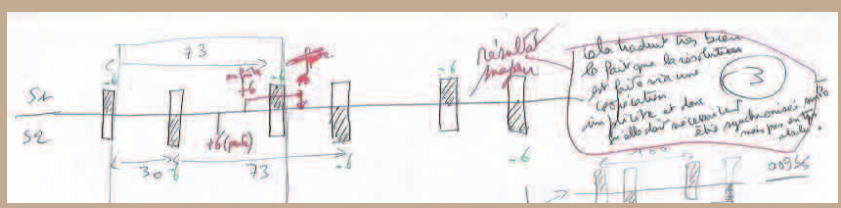


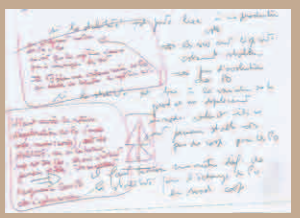


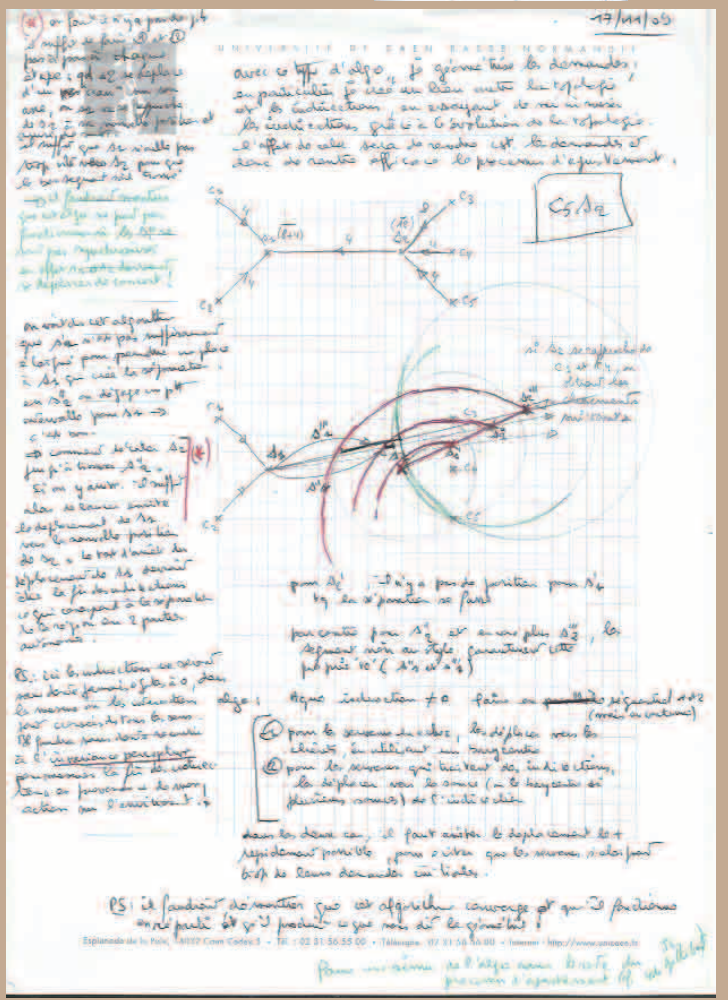


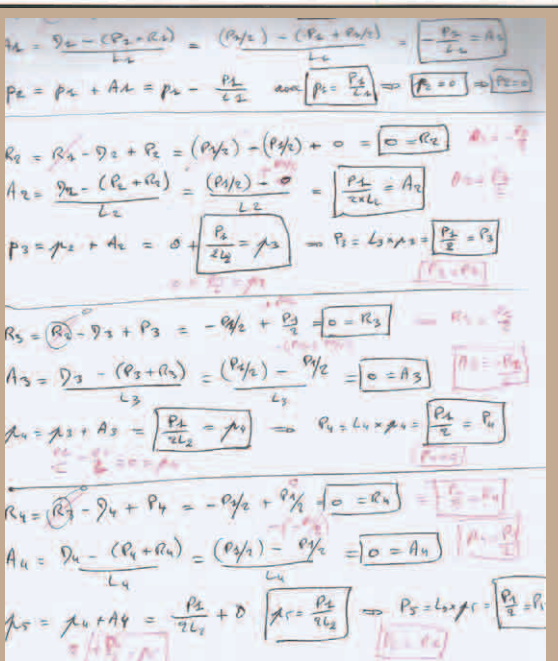








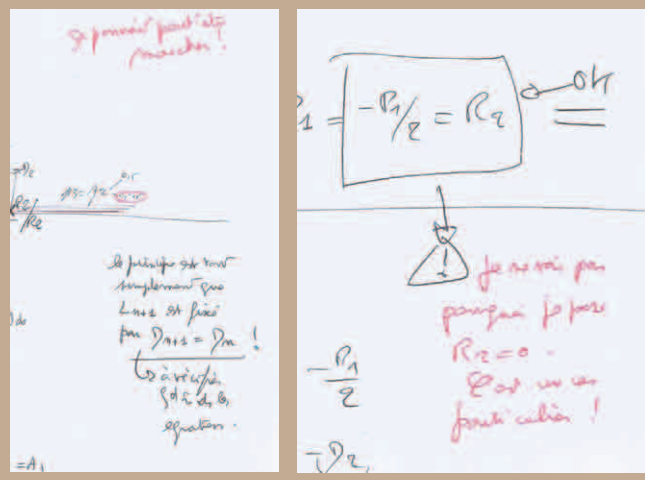


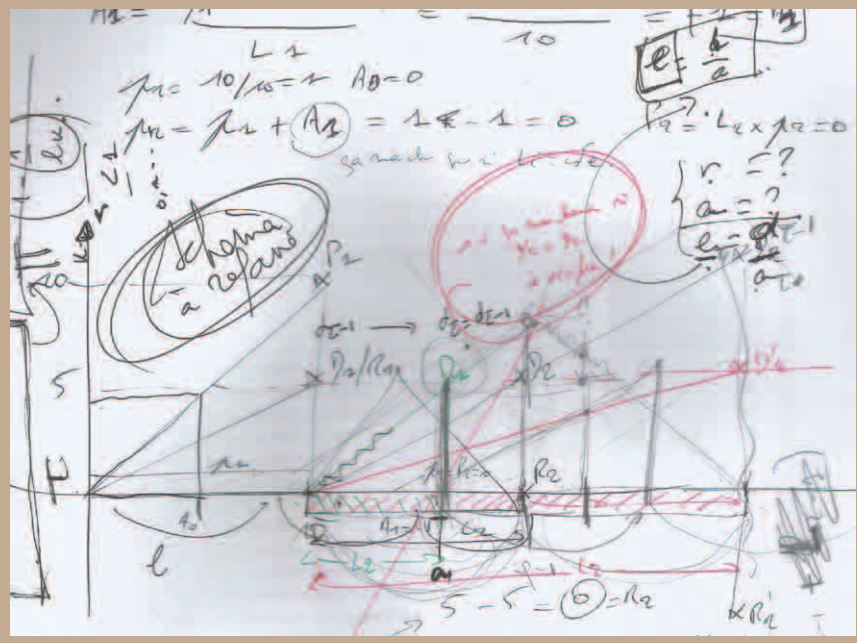


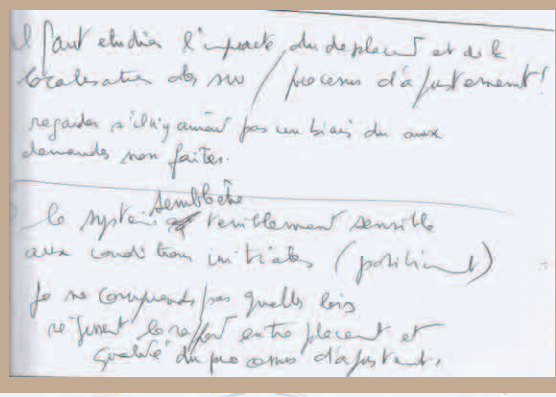
Je dois reconnaître que le rouge permet de mettre en avant des annotations graphico-textuelles intéressantes.
J'ai aussi utilisé plusieurs couleurs pour détacher des plans de commentaires sur des écrits, comme le vert ci-contre.

Peut-être n'en ferions-nous pas du papier peint, mais je trouve qu'il se dégage quand même une certaine poésie dans la régularité de ces formules…

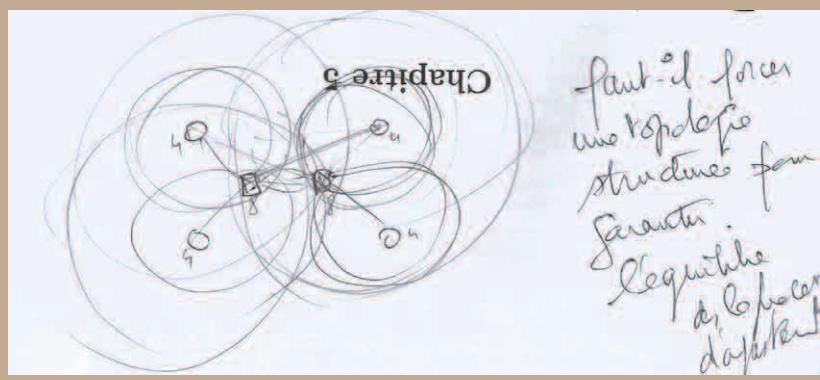
En pleine discussion avec moi-même ! C'est une très belle métaphore de la communication par le contexte, si chère aux systèmes multi-agents auxquels je suis attaché, pour toujours, non pas par immobilisme, moi qui ait toujours défendu la mobilité comme une composante cognitive élémentaire, mais parce que j'y crois dur comme fer !
Et là, par cette phrase à rallonge, je fais un hommage, sans doute pas très honorable mais absolument sincère, au style de l'un de mes écrivains préféré qui est Joseph Conrad.



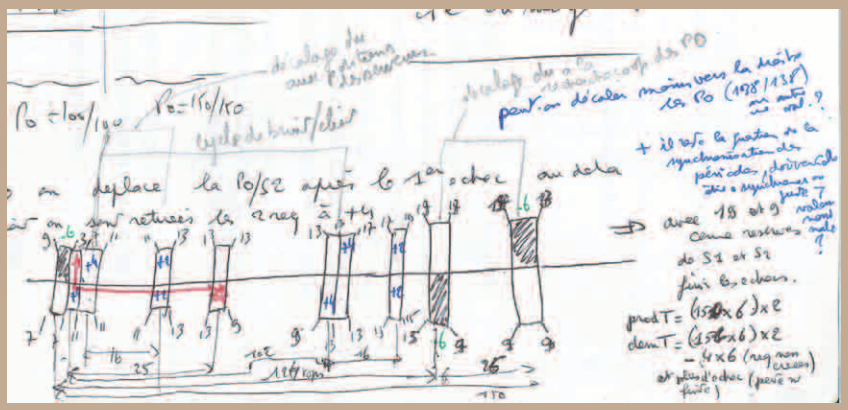


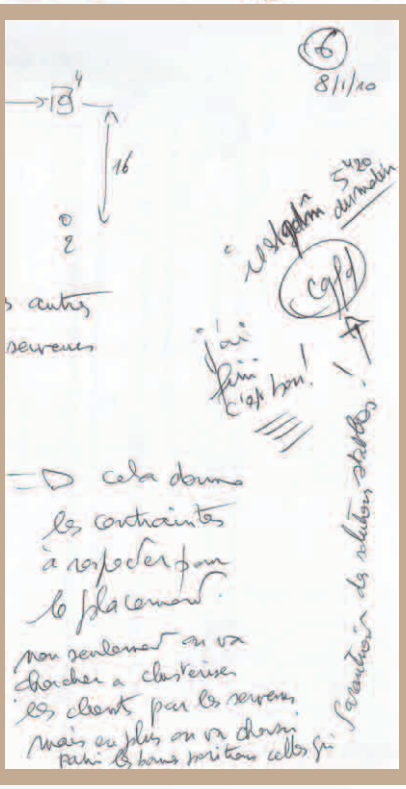


La pensée se heurte à des fragments polymorphes pas toujours compréhensibles ex abrupto.

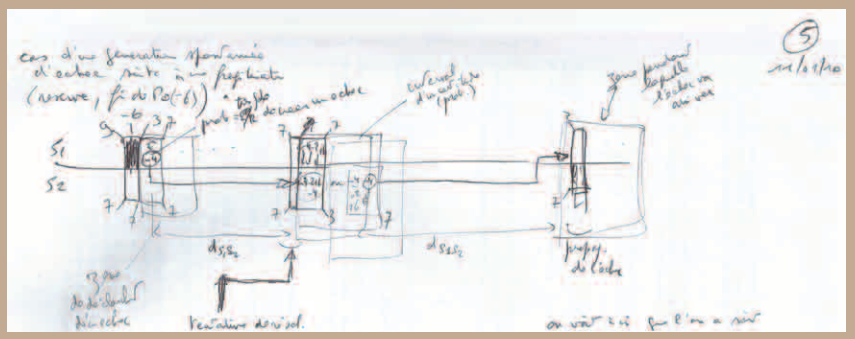


2010

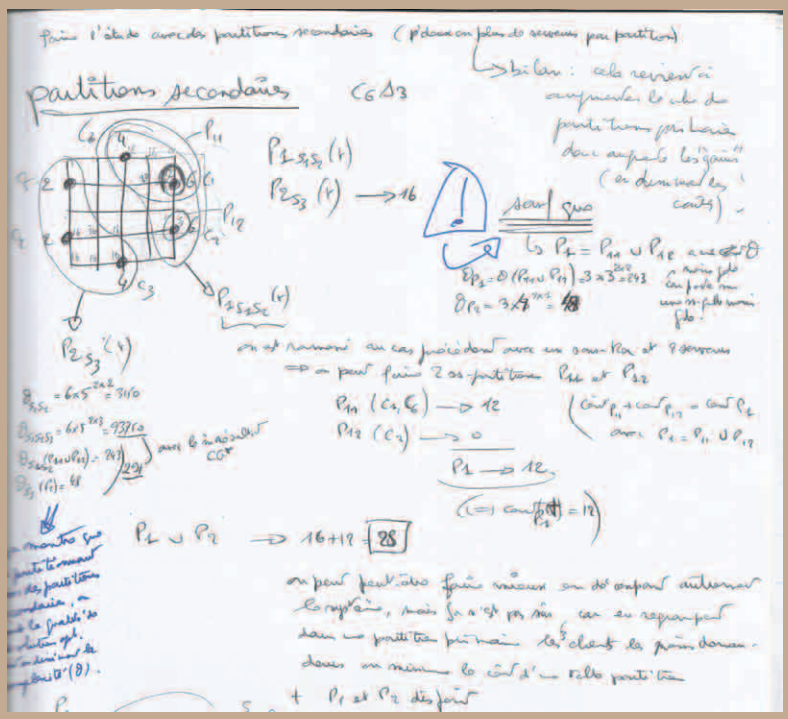


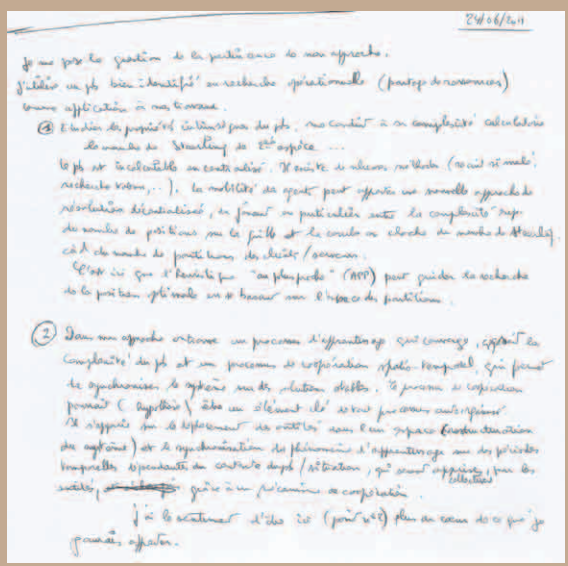


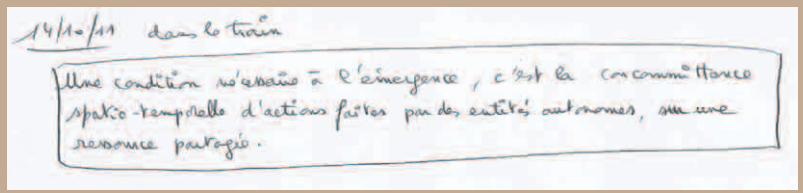
Un moment haut !



2011





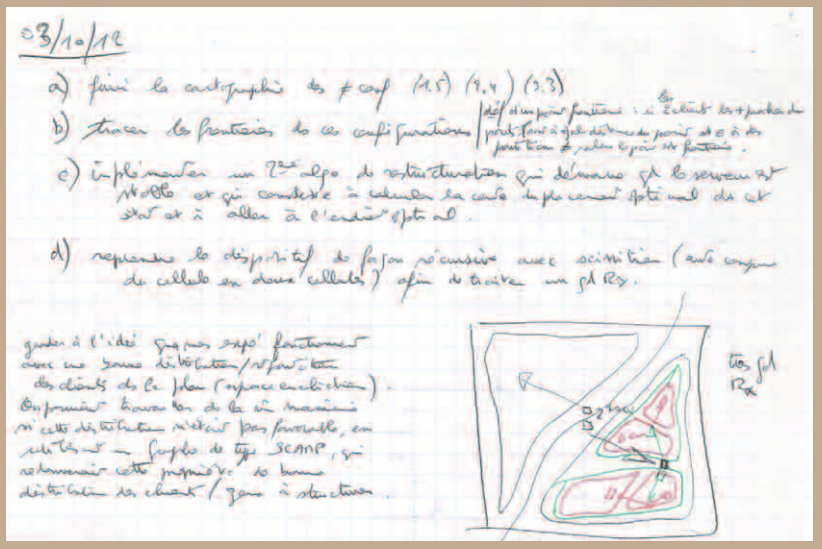


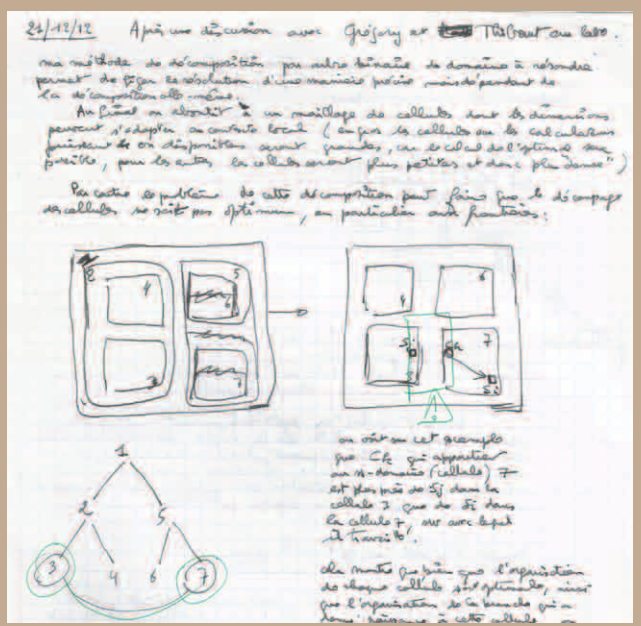
Et toujours ces questionnements !

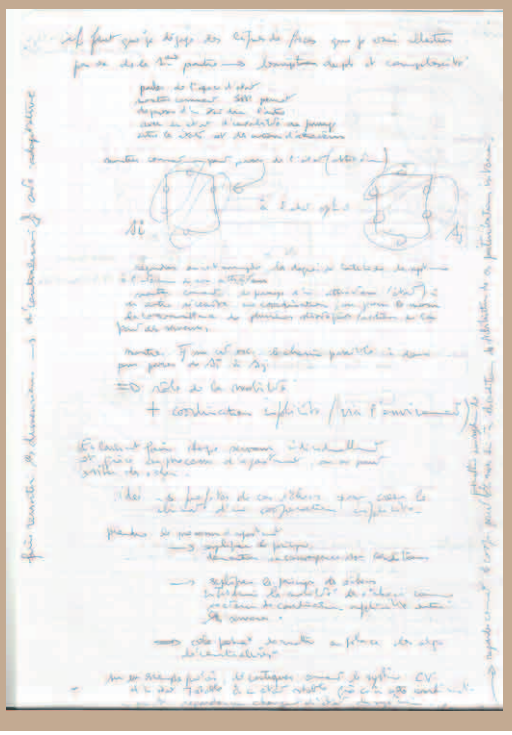
Cette injonction rassurante… plus qu'éclairante !

2012

Un peu de méthode dans ce monde de brume !







2014

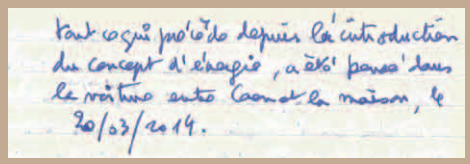
Spatialiser ses lieux de pensée, de surcroît à un moment précis, c'est plus newtonien que quantique. De là à justifier l'importance de la dérive des connaissances dans l'espace de représentation des connaissances, il n'y aurait qu'un pas que je pourrais franchir…

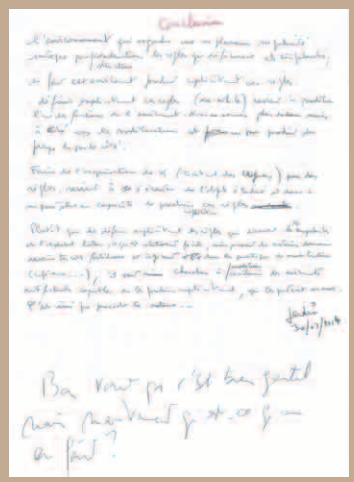
Encore une injonction au doute !

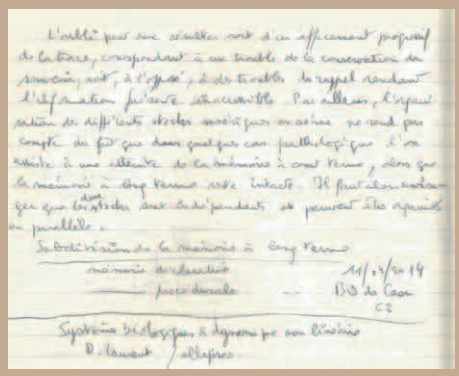
Ha, l'anthropomorphisme mémoriel…
ça donne des idées !

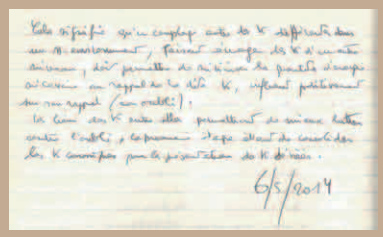
Avec les lettres de l'alphabet qui prennent de plus en plus de place dans les connaissances basiques à représenter !

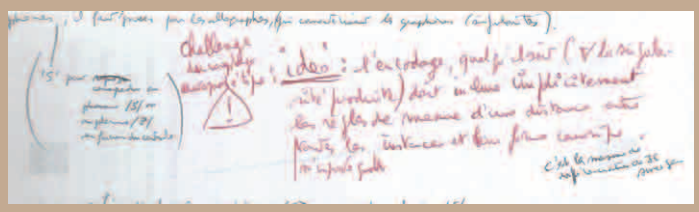
Le cheminement de la planche ci-contre est intéressant. En effet je pars d'un algorithme (à gauche), qui se transforme en une idée (en bas à gauche), puis en un pari (au centre) pour finir en un plan opérationnel (à droite) : je dispose, je suppute, je joue et je m'y colle !

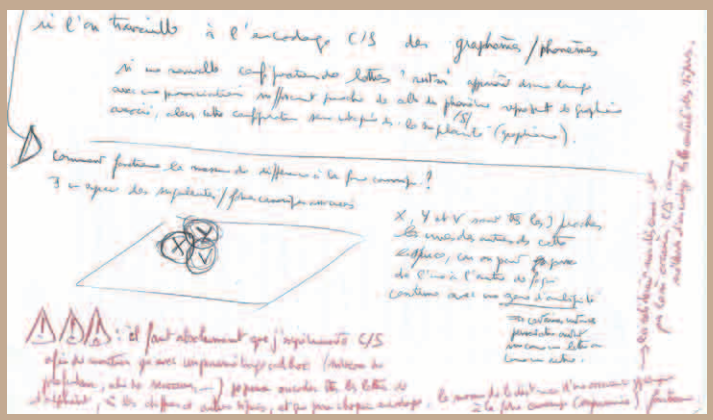


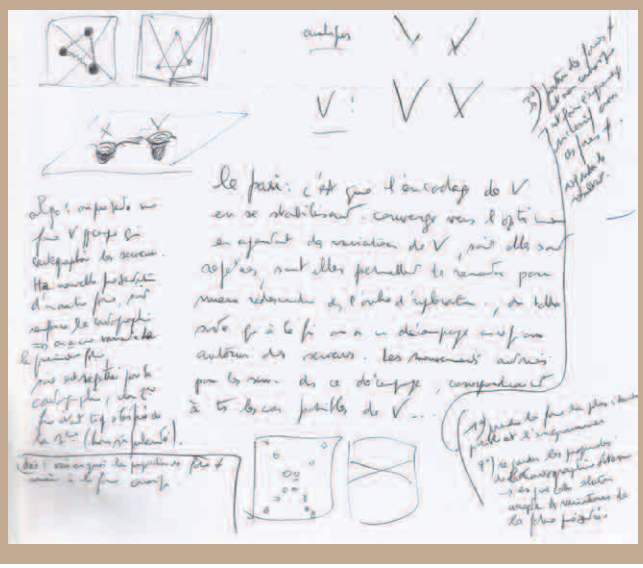


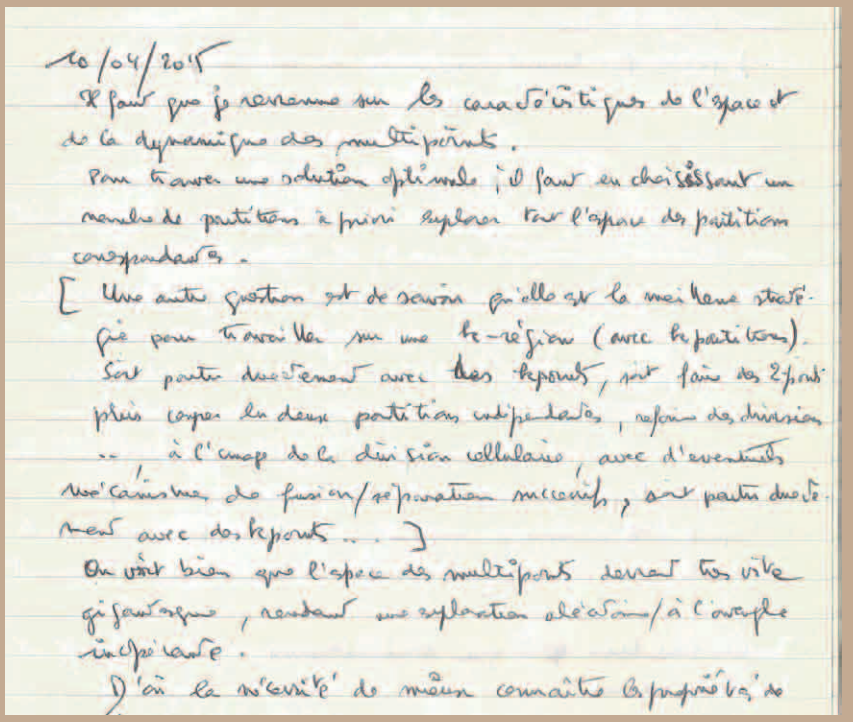












10/04/2015

Il faut que je reprenne sur les caractéristiques de l'espace et de la dynamique des multipoints.

Pour trouver une solution optimale, il faut en choisissant un nombre de partitions à priori explorer tout l'espace des partitions correspondantes.

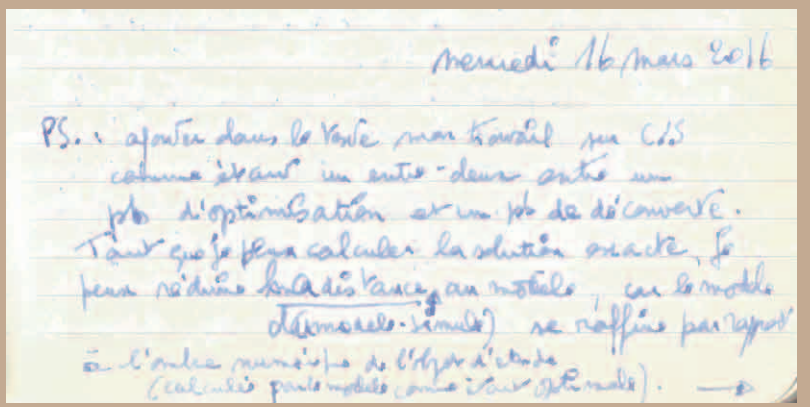
[Une autre question est de savoir quelle est la meilleure stratégie pour travailler sur une ré-région (avec repartition). Soit partir directement avec des reports, soit faire des 2 partitions (coupe la deux partitions indépendantes, refaire des divisions à l'image de la division cellulaire, avec d'éventuels mécanismes de fusion/séparation nécessaires), soit partir directement avec des reports ...]

On voit bien que l'espace des multipoints devient très vite gigantesque, rendant une exploration aléatoire/à l'aveugle inespérable.

D'où la nécessité de mieux connaître la géométrie de

2015

En arrière toute! Cette injonction va conduire, quelques années plus tard à l'écriture du chapitre III, du présent document, qui théorise les multi-points et leur dynamique associée.

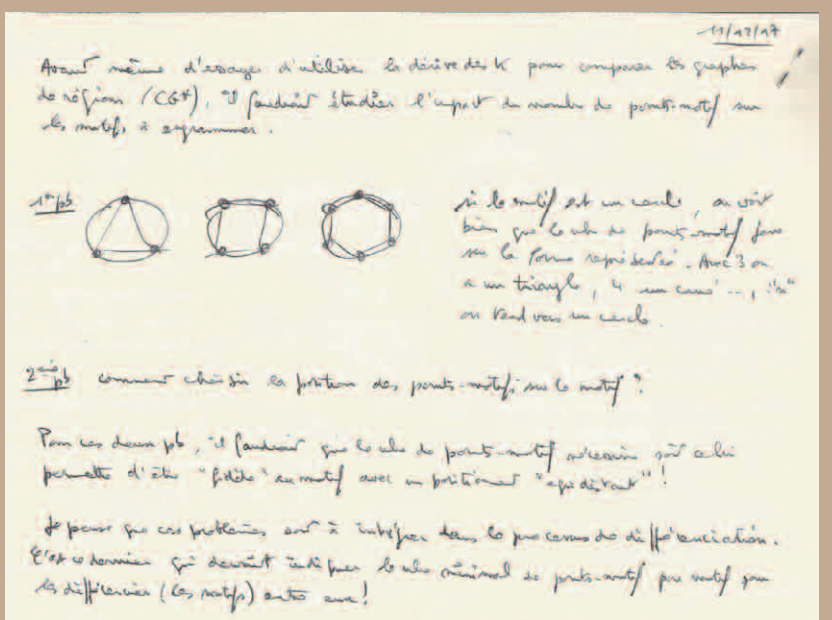


Mercredi 16 mars 2016

PS : ajouter dans le texte mon travail sur CdS comme étant un entre-deux entre un pb d'optimisation et un pb de découverte. Tant que je peux calculer la solution exacte, je peux rédiger les variables au modèle, car le modèle (déterministe-simulé) se réaffine par rapport à l'ordre numérique de l'objet d'étude (calculé par le modèle comme étant optimal).

2016

Mon aboutissement final du raffinement du modèle est arrivé avec la v6 (MVB) des simulations, développée au chapitre IV.



11/11/17

Avant même d'essayer d'utiliser la théorie des K pour comparer les graphes de région (CdS), il faudrait étudier l'aspect du nombre de points-motifs sur les motifs à segmenter.

1er pb

si le motif est un carré, on voit bien que le nbr de points-motifs forme la forme représentée. Avec 3 on a un triangle, 4 un carré, ... on tend vers un cercle.

2ème pb comment choisir la position des points-motifs sur le motif ?

Pour ces deux pb, il faudrait que le nbr de points-motifs nécessaire soit celui permettant d'être "fidèle" au motif avec un positionnement "optimisant" !

Je pense que ces problèmes sont à intégrer dans la problématique de différenciation. C'est le nombre qui devient intéressant le plus minimal de points-motifs par motif pour les différencier (les motifs) entre eux !

2017

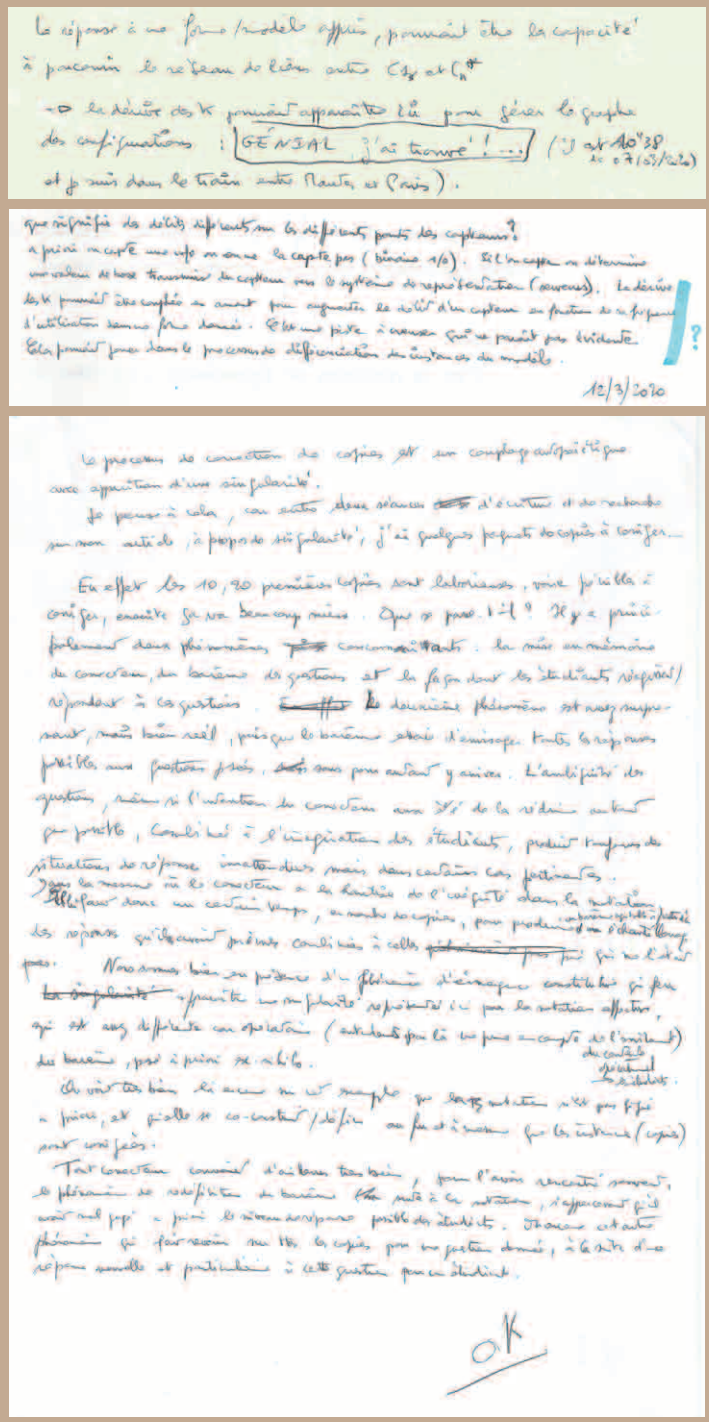
Encore une piste toujours prometteuse…

2020

Phase haute ; comme ça fait du bien !

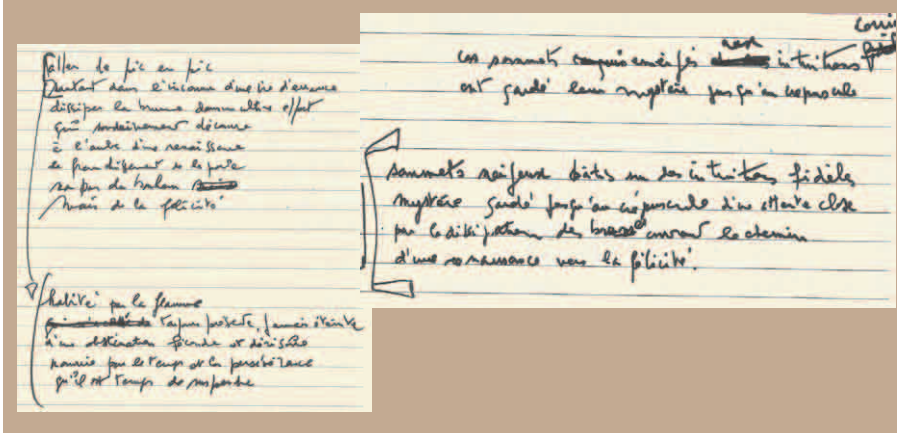
Et cinq jours plus tard, phase descendante avec possibilité d'amélioration…

Ou comment faire de l'anthropomorphisme pédagogique avec ses préoccupations de recherche ?
Ceci dit, je suis assez convaincu de la justesse du phénomène, tel que je le décris et que j'ai rencontré pendant toute ma carrière d'enseignant à l'université.
Corriger des copies est un phénomène quantique, que l'on pourrait formuler, a priori de façon naive, comme : c'est en corrigeant que l'on corrige !
En effet l'évaluation passe par un questionnement fait aux étudiants, dont les réponses influencent les questions elles-mêmes, par le biais du barême. Le meilleur exemple de ce phénomène est celui du sorcier sur le pont dans le film «Sacré Graal» des Monty-Python, où ce dernier pose une question sur les hirondelles au preux chevalier, qui veut emprunter le dit pont ; une bonne réponse conditionne son passage. La réponse du chevalier en forme de question (Vous voulez parler des hirondelles d'Afrique ou d'Asie…?) renvoie au sorcier que sa question est mal formulée et c'est lui qui disparaît…

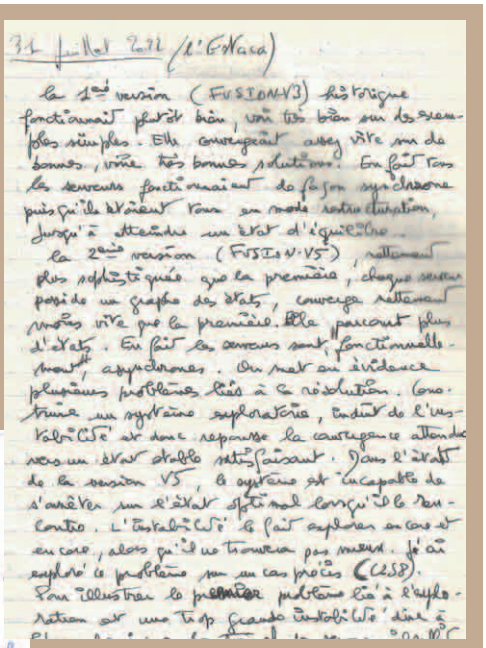


2021

Même la poésie la plus naïve conduit son auteur vers une félicité apaisante.

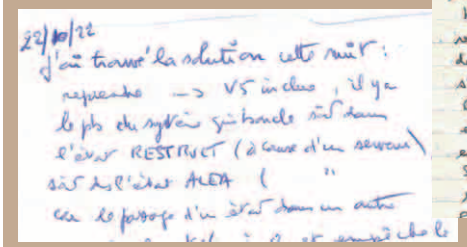


2022


31 juillet 2022 (l'Estaca)

La 1ère version (FUSION-V3) historique fonctionnait plutôt bien, vu très bien sur des exemples simples. Elle convergeait assez vite vers de bonnes, voire très bonnes, solutions. En fait tous les serveurs fonctionnaient de façon synchrone puisqu'ils devaient tous en mode restriction, jusqu'à atteindre un état d'équilibre.

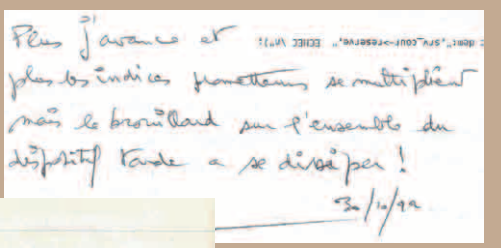
La 2ème version (FUSION-V5), nettement plus sophistiquée que la première, chaque serveur possède un graphe des états, converge nettement moins vite que la première. Elle parcourt plus d'états. En fait les serveurs sont fonctionnellement aguerris. On met en évidence plusieurs problèmes liés à la résolution. Construire un système explorateur, tendre de l'universalité et donc repousser la convergence attendue vers un état stable satisfaisant. Dans l'état de la version V5, le système est incapable de s'arrêter sur l'état optimal lorsqu'il le rencontre. L'instabilité fait explorer encore et encore, alors qu'il ne trouvera pas mieux. J'ai exploré le problème sur un cas précis (CAS8). Pour illustrer le premier problème lié à l'exploration et une trop grande instabilité dans


22/10/22

J'ai trouvé la solution cette nuit :

reprendre → V5 inchangé, il y a le pb du système qui bascule sur dans l'état RESTRICT (à cause d'un serveur) sur de l'état ALEA (

car le passage d'un état dans un autre

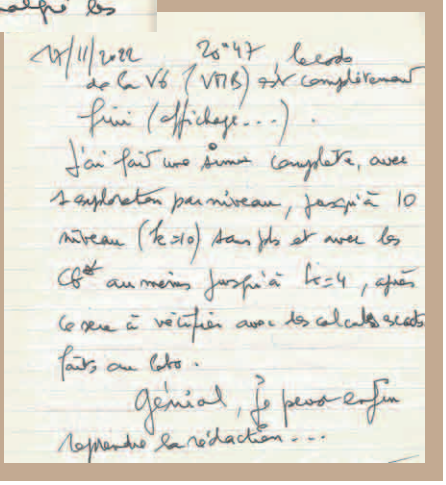

Plus j'avance et plus les indices prometteurs se multiplient mais le brouillard sur l'ensemble du dispositif tarde à se dissiper !

3e/10/22


8 novembre 2022

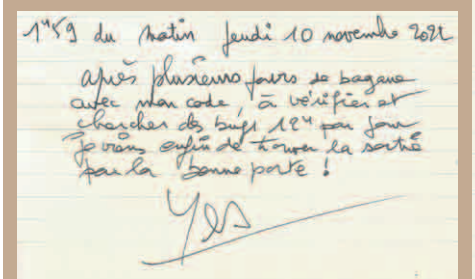
Je commence à pouvoir mettre des mots sur mon travail. Depuis très longtemps je suis fasciné par l'émergence de comportements dus à des agrégats de comportements individuels. Les nids d'oiseaux en sont un exemple emblématique pour moi car il a été abordé dans le monde de la recherche informatique. Bien d'autres existent, à commencer par ceux issus de sociétés d'humains et autres (animaux, plantes, ...). Un autre aspect m'a toujours fasciné, c'est celui de la représentation des connaissances. Comment faisons-nous au fond pour manipuler des connaissances, qu'elles soient symboliques, sensorielles ou autres... ? Ces deux pôles m'ont donc guidé tout au long de mes années d'errance et de traîne la plume sur la feuille blanche, plus ou blanche du reste !

Quelques rencontres auront été décisives dans ma quête sans cesse poursuivie malgré les


28/11/2022 20h47, le code de la V6 (V6B) est complètement fini (affichage...).

J'ai fait une simu complète, avec 1 exploration par niveau, jusqu'à 10 niveau ($k \geq 10$) dans les et avec les CGT au moins jusqu'à $k=4$, après le reste à vérifier avec des calculs exacts faits au fato.

Génial, je peux enfin reprendre la rédaction...


1h59 du matin jeudi 10 novembre 2022

Après plusieurs jours de bagarre avec mon code, à vérifier et chercher des bugs 1h24 par jour je vais enfin de trouver la sortie par la bonne porte !

YDS

Un coup de rétroviseur salvateur, qui va donner la version v6 (MVB).

Est-ce de l'inconscience, de la subconscience ou de l'inertie consciente ?

Un constat, une direction (comme la flèche du temps), un jeu de piste qui se densifie et une météo clémente qui se fait attendre, voire désirer !

Pourquoi dire la même chose avec le même nombre (environ) de caractères, mais autrement ? Est-ce une clarification ou un obscurcissement ?

Espèce de traîne-la-plume !

Encore le recours à l'anthropomorphisme, comme boué de sauvetage pour exprimer tout le respect que j'ai du «Code», cette activité si particulière, qui peut s'avérer extrêmement retors.

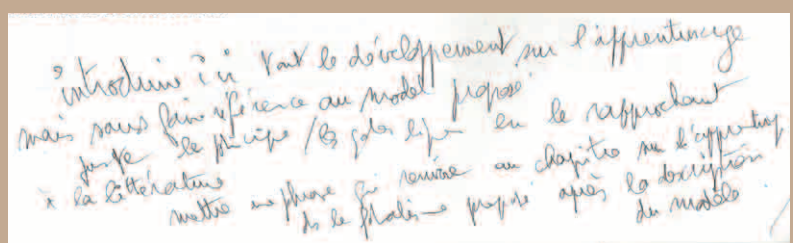
2023

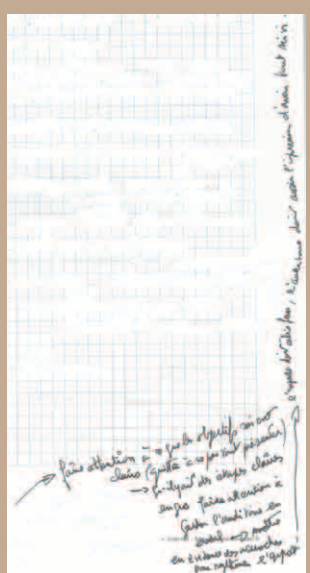
Les connaissances sont-elles :

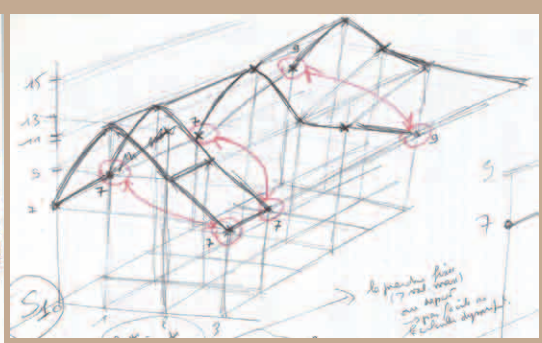
- *spatio-temporelles ?*
- *dans un espace ou dans plusieurs simultanément ?*
- *performatives ?*
- *…*

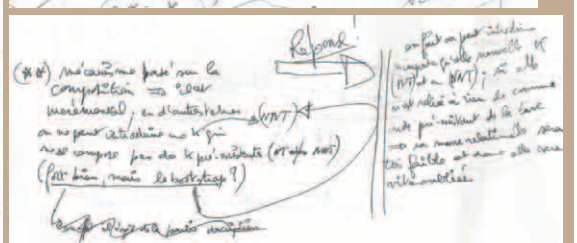
????

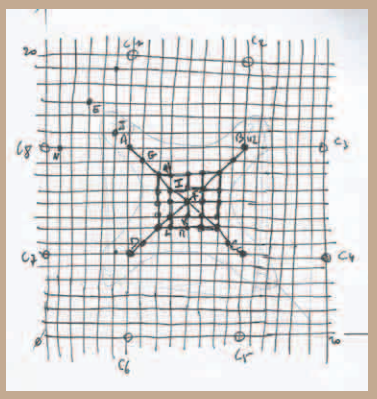
Ces extraits, de mes carnets de notes, n'ont pas été datés précisément comme les précédents. Néanmoins ils ont été faits entre la thèse et l'HDR donc entre 1992 et 1998.

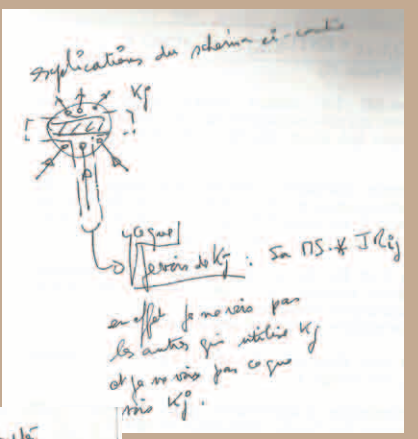


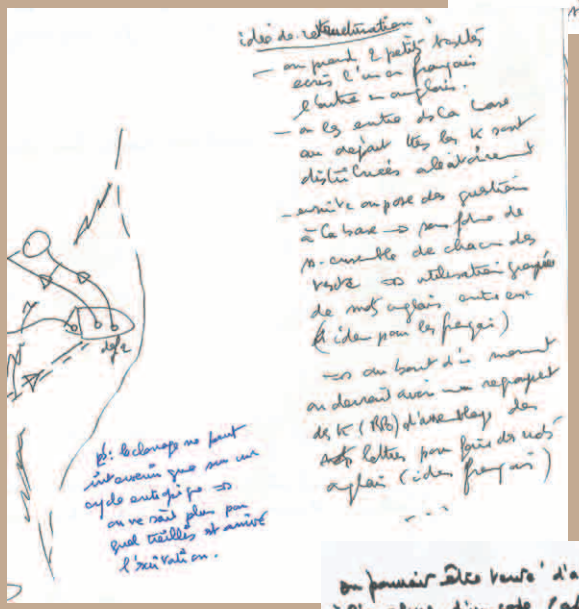


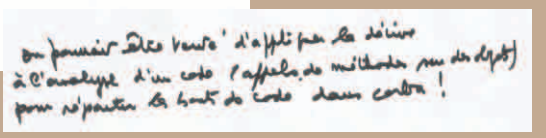


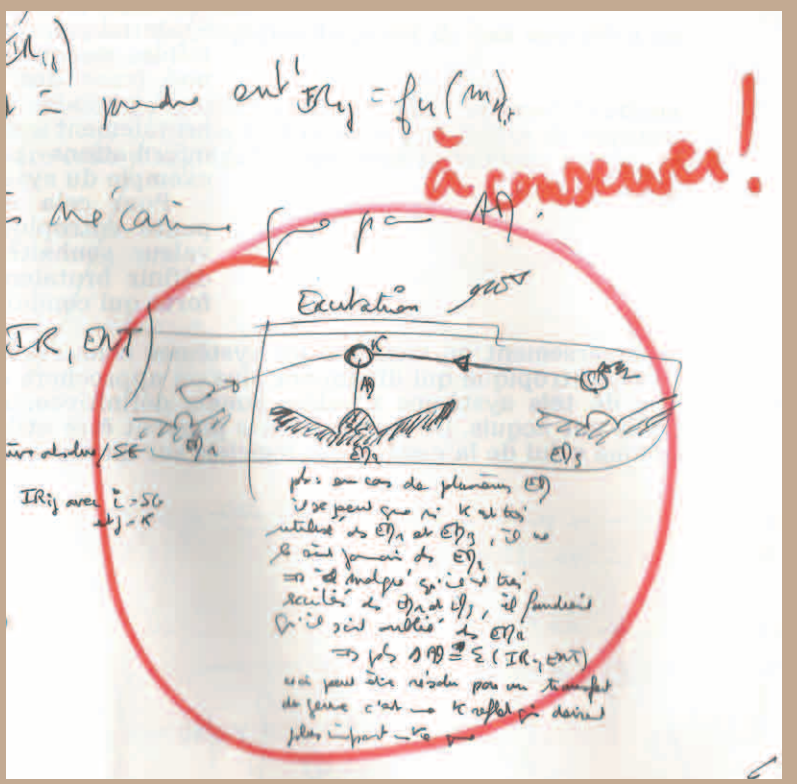












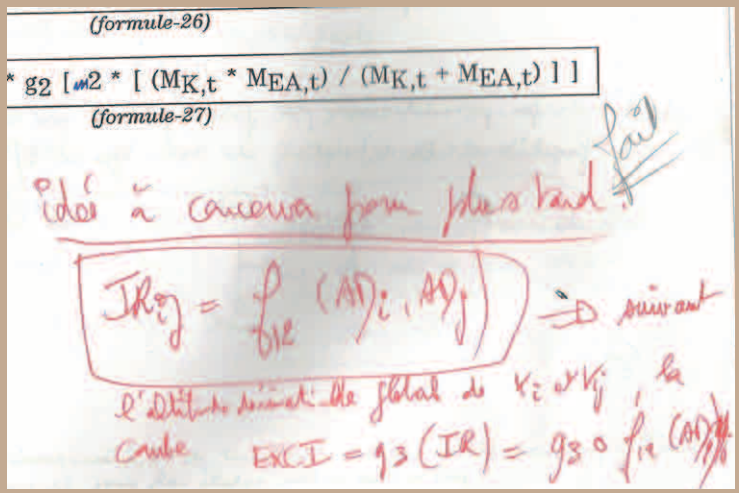
Et encore des idées…

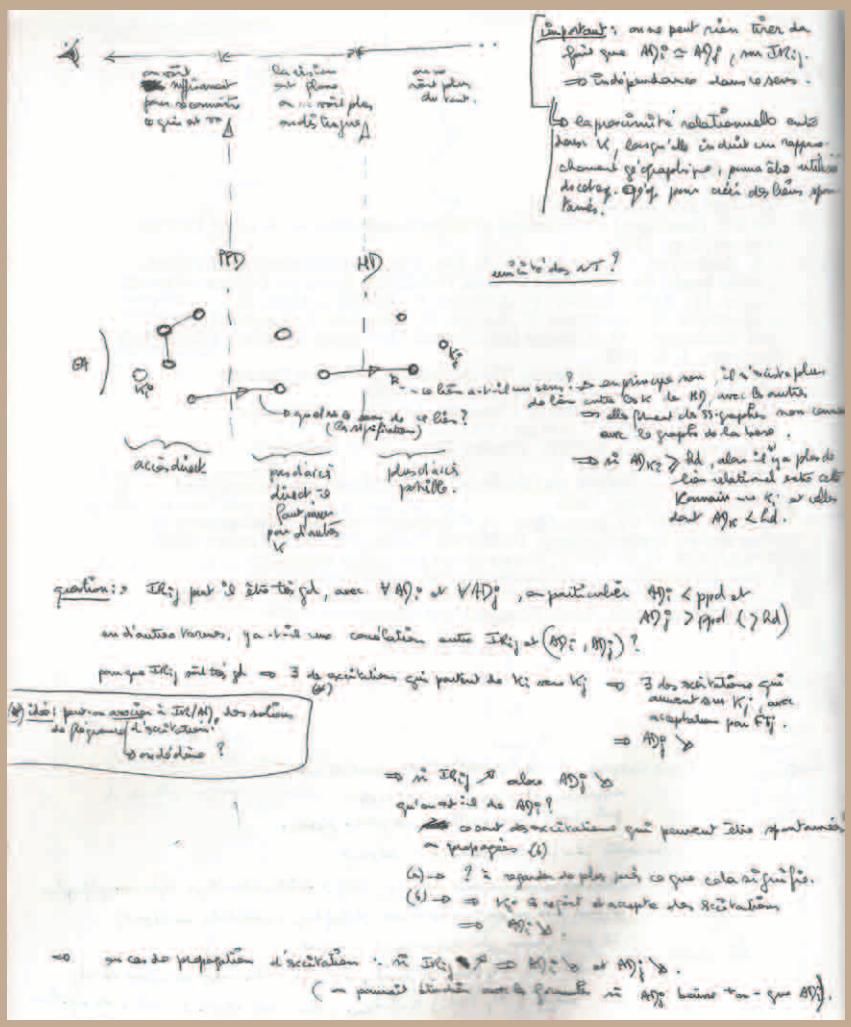
Ici je fais allusion à l'usage du modèle de dérive des connaissances dans la programmation répartie CORBA. Avec cette technique, assez géniale à mon sens, dont j'ai été parmi l'un des derniers à l'enseigner en école d'ingénieur, on peut déplacer des objets à travers le réseau et les invoquer de n'importe quel endroit de ce dernier. Cette proposition aurait permis de déplacer les classes (le code), correspondant aux invocations, de façon judicieuse vis-à-vis de ces dernières, sachant que de nombreux objets distants ont recours à ces classes.
Cela revenait, ni plus ni moins, à appliquer C_nS_k à cette problématique du code partagé, avec comme ressources partagées ces fameuses classes !

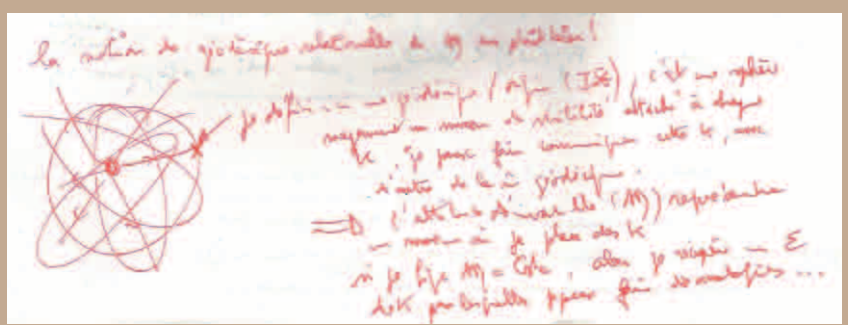
Oui, mais comment ?
Dans un testament, peut-être ?

Beaucoup de questions !

Et pour finir en beauté, pourquoi ne pas se faire plaisir avec une géodèsique relationnelle ?
Là encore, l'espace spatio-temporel a explosé, ne pouvant dater précisément cette planche !







ÉPILOGUE ET REMERCIEMENTS

X

Nous voilà arrivé à la fin de ce voyage. J'ai consacré volontairement, dans ce document (dans la partie IV-D du chapitre IV), une part belle à l'action de coder pour produire le Code, acteur indispensable dans nos réflexions. Non seulement j'ai décrit par le menu le code de la dernière version (MVB) de nos simulations, afin qu'il puisse être compris, voire réécrit sous d'autres horizons, mais je me suis également attaché à décrire les versions précédentes (de la v3 à la v5) pour mieux faire comprendre la dynamique du processus général qui aura nourri ce travail.

La première partie de cet épilogue me donne l'occasion de faire un petit éloge de l'action de coder, une activité qui demande beaucoup d'entraînement, mais qui, malheureusement ou heureusement, nécessite des qualités intrinsèques.

La seconde et dernière partie de ce testament ne pouvait pas ne pas rendre hommage à toutes les personnes nombreuses qui ont joué un rôle plus ou moins déterminant, plus ou moins soutenu, plus ou moins bienveillant…, à ma carrière de chercheur. C'est ce que je vais m'employer à faire sans hiérarchie, ni chronologie, tout en sachant que certaines personnes vont être oubliées, de façon tout-à-fait involontaire. Je m'en excuse par avance.

Petit éloge du code

X-A

Je ne peux pas refermer ces pages sans célébrer quelque chose qui, d'une certaine façon, participe à la noblesse de la profession d'informaticien.
Je veux parler de l'action de coder, écrire du code, faire des programmes, en un mot le codage.

Au fil des années la question du codage est devenue paradoxale, le chaud et le froid, le haut et le bas, l'élégance et le machinal, le blanc et le noir…

J'évacue tout de suite les notions de gain ou de position sociale. Dans la galaxie informatique, coder vous placera soit tout en bas de l'échelle, soit tout en haut !
Inutile d'insister sur le fait, qu'à l'image de l'organisation pyramidale de la mémoire dans un ordinateur (on a les comparaisons que l'on peut…), la très grande majorité des informaticiens codeurs est en bas. Les étages intermédiaires sont de plus en plus occupés par des outils, ou programmes informatiques, qui «codent» à leur façon, facilitant la vie des uns et des autres, y compris celle de non-informaticiens.

Au-delà de l'image commune du geek, que je ne suis pas et n'ai jamais été, je voudrais, à travers mon expérience, saluer l'aventure du codage qui m'aura accompagné durant toutes ces années professionnelles.

L'affaire était pourtant mal engagée, mais moins mal que pour ceux qui, comme mon père, en sont restés aux premières calculatrices quatre opérations. Après un Deug de math-physique obtenu en 1979 à l'Université de Caen, je suis allé à l'Université Paris-VI à Jussieu, pour entamer une Licence d'informatique, formation inexistante à l'époque à Caen.

Nous n'avions pas ou peu d'ordinateur personnel (PC). Bien qu'étant dans la plus grande faculté de France, nous devions programmer en tapant chaque ligne de nos programmes sur des machines à perforer des cartes, placer le tas de cartes dans des cartons mis à la disposition des étudiants. Une carte rose devait séparer notre programme (cartes jaunes) de ceux de mes camarades. En Licence nous étions plus de deux-cents étudiants. Le lendemain nous récupérions, dans notre casier, le listing du résultat de l'exécution de notre programme perforé.
Inutile de dire que, dans ces conditions, nous étions de piètres programmeurs. Disons que l'apprentissage était laborieux et pas très enthousiasmant.

Arrivé en DEA (l'équivalent du Master-2 recherche actuel) en 1981, les choses se sont nettement éclaircies. Les plus chanceux, dont je ne faisais pas partie, avaient leur PC, pendant que les autres travaillaient, dans de grandes salles, sur des écrans, en libre service, qui étaient reliés par Renater à un gros calculateur installé sur le campus de Rennes.

C'est dans ces conditions que j'ai rencontré, pour la première fois, le langage 'C' qui arrivait tout juste en Europe, après avoir été inventé en 1972 par Dennis Ritchie

et Ken Thompson dans les laboratoires Bell. Ce langage a été stabilisé et rendu populaire en 1978 par Brian Kernighan, qui fut le principal auteur du livre "*The C Programming Language*".

J'ai eu l'occasion de pratiquer différents types de langages, en passant par des langages dédiés à l'I.A. comme le *Lisp* ou *Airelle*, un langage objet fondé sur *Lisp* et inventé par des collègues de l'Université de Caen. Je dois avoir été l'un des premiers à avoir acheté une licence d'*Airelle*, à l'époque où je travaillais au S.E.P.T.

J'ai beaucoup utilisé le langage d'*oRis* pour mes simulations. C'est un langage objet qui est interprêté$^{++}$. Il est basé sur le *C*$^{++}$ (cf. les parties IV-C1 et IV-C2 au chapitre IV).

Parallèlement à ces expériences en recherche j'ai enseigné pendant de nombreuses années le 'C', en allant dans ses retranchements les plus lointains et les plus excitants. Il faut dire que ce langage vous permet de jouer avec la mémoire comme aucun autre langage. C'est un véritable terrain de jeu pour les programmeurs virtuoses. Ceci explique son exceptionnelle longévité dans des milieux «très éclairés», même si un grand nombre d'entre eux se trouvent dans le Darknet.

Au-delà de cette introduction un peu longue, qui a le mérite de planter mon décors, je voulais dire quelques mots sur le codage, comme je l'ai vécu.

Coder signifie développer des programmes qui peuvent se déployer dans l'espace et dans le temps.

Là encore, je mets volontairement de côté toutes les méthodologies et les outils associés (framework…), qui fleurissent de plus en plus pour mettre l'informatique et la réalisation de programmes à la portée du plus grand nombre.

Le «codage» est une activité fondamentalement quantique qui prend en compte simultanément un très grand nombre de chemins possibles pour aller à un point donné, qui est l'objectif du programme visé. Il existe d'ailleurs des travaux de recherche qui font de la preuve de programme, dont le but est de parcourir tous ces chemins tout en vérifiant certaines propriétés (terminaison, cohérence…).

Il faut ajouter à cette complexité intrinsèque un nombre incalculable, pour dire très grand, de paramètres intervenant dans le déroulé de ces différents chemins possibles.

Le programmeur est le chef d'orchestre de cette aventure qui doit avoir une vision, dans l'espace et dans le temps, la plus large et la plus précise possible de tout cet imbroglio. Comme je l'avais formulé dans mon «moulin à légumes», jadis à la conférence TOOLS'1990, il faut procéder par étapes successives qui consistent à injecter du code dans l'existant après être sûr du bon fonctionnement de ce dernier et sans vouloir être trop gourmand!

C’est particulièrement délicat lorsque le code est très imbriqué, c.-à-d. où tout dépend de tout, et que des ressources sont partagées par des fils d’exécutions parallèles. Si en plus on ajoute une bonne dose de récursivité…

Qui n’a pas fait l’expérience[1] de l’ajout d’une ligne de code innocente qui a fait explosé en vol tout le programme beaucoup plus loin, c.-à-d. dans l’espace du code et dans les multiples ramifications temporelles de son exécution.

L’un des plus beaux et des plus complexes codes qui existe est celui du système d’exploitation *Linux*, fondé en 1991 par Linus Torvalds. En 2020 le noyau comptait 27.8 millions de lignes (incluant le code, les fichiers de configuration, les commentaires,…).

À ma modeste échelle, c’est de cette façon incrémentale que j’ai procédé entre les différentes versions de mes simulations.

Dans ces conditions il existe un plafond de verre, propre à chaque programmeur, au-delà duquel ses capacités de maîtrise de la complexité grandissante, deviennent insuffisantes. J’ai commencé à entr’apercevoir mon plafond avec MVB.

Même si le codage n’est qu’un moyen pour arriver à une fin, il n’en demeure pas moins que le voyage est intense et tout-à-fait remarquable.

[!s.f]

1 L’un de mes exercices préférés, dans mon cours de Systèmes en ‘C’, consistait à provoquer une réaction en chaîne qui illustrait parfaitement l’effet papillon. Il sufisait que l’étudiant déplace une ligne anodine d’envoi de signal, ce que mes étudiants ne tardaient jamais à faire, pour provoquer une cascade de réactions allant jusqu’à la fermeture de la session ouverte et l’arrêt de tous ses processus ! L’effet était garanti, surtout quand l’étudiant, cela arrivait par moments, avait ouvert des fenêtres, dans sa session, qui n’étaient que très moyennement en rapport avec mon cours… Passé l’effet de surprise, nous regardions ensemble la cascade d’évènements en jeu.

Remerciements

X-B

Toute cette carrière n'aurait pas été la même si je n'avais pas rencontré de nombreuses personnes dans des occasions aussi diverses que variées. Sans aucune chronologie et en espérant ne pas oublier trop de personnes, voici toutes celles que je souhaite remercier ici.

En tout premier lieu mes collègues du département informatique de l'I.U.T. de Caen, et plus particulièrement Eric Porcq, qui m'ont permis d'abandonner mes responsabilités administratives pendant mes deux dernières années d'enseignement, afin de me consacrer exclusivement à ce testament, en plus de mes cours. Eric, qui m'avait accompagné lors de la création de la licence professionnelle ASRSI, a repris très spontanément la responsabilité de cette formation.

J'ai une pensée émue pour mon directeur de thèse, Martial Vivet, qui m'a fait confiance à un moment où ma situation hors du giron académique n'était pas simple.

Je remercie également Bernard Victorri qui a eu la patience de m'écouter à mi-parcours de ma thèse, à un moment où j'étais un peu perdu dans mes réflexions, et surtout d'accepter de la rapporter un an et demi plus tard.

Comment ne pas avoir également une pensée émue pour Jean-Marc Fouet, que je ne connaissais pas à l'époque, en dehors de ses productions scientifiques. Quand mes collègues du groupe de thésards suivis par Martial Vivet, ont su que je l'avais choisi pour être mon second rapporteur, ils me dirent que je ne n'avais peut-être pas fait le meilleur choix. Jean-Marc Fouet avait une réputation d'être extrêmement cassant. J'ai pu le vérifier plus tard lorsque Fabrice Harrouet et Roger Cozien présentèrent le simulateur *oRis*. Il était au fond de la salle et était très, très remonté contre ce logiciel. Jean-Marc a beaucoup aimé mon travail (cf. la fin de son rapport de thèse dans la partie X-A). Il n'est malheureusement plus là pour voir ce que je suis devenu. Je me demande bien ce qu'il aurait pensé de la suite, sachant que j'ai beaucoup utilisé *oRis* ?

Merci également à Fabrice Harrouet et Roger Cozien justement pour la réalisation d'*oRis* (novembre 2000), un logiciel merveilleux, qui n'aura pas eu la reconnaisance qu'il mérite et qui m'aura rendu de très grands services. Je dois être le dernier à l'utiliser. Même son concepteur l'a abandonné depuis longtemps !

Un grand merci aussi à mon collègue Serge Mauger, avec qui nous avons eu de longues discussions passionnantes. Je lui dois mon ouverture à la linguistique, même si elle reste modeste, elle m'aura permis d'avancer dans les concepts de représentation des connaissances, tout spécialement autour des lettres de l'alphabet.

Je remercie Pascal Boldini, chercheur au laboratoire d'I.A. et de Sciences Cognitives de l'ENST-Bretagne à Brest, pour m'avoir accompagné dans mes réflexions sur l'apport, dans mon travail, de la théorie des catégories en mathématiques. L'interdisciplinarité n'est pas quelque chose de si courant en sciences; cela vaut la peine de signaler des expériences qui vont dans ce sens.

Je remercie les membres de l'équipe «Modèles, Agents, Décisions» (MAD) que j'ai créé au laboratoire GREYC à l'Université de Caen en septembre 2002. En particulier Abdel Illah Mouaddib qui a repris et développé avec beaucoup de réussite cette équipe, en lui donnant une très forte coloration autour des chaînes de Markov. Je remercie également Bruno Zanuttini et Grégory Bonnet qui ont repris à leur tour le flambeau, perpétuant ainsi cette très belle équipe de recherche. C'est aussi grâce à eux que j'ai pu être chercheur-associé au GREYC, pendant les deux dernières années de réalisation de ce testament, ce qui m'a soutenu moralement pour aller jusqu'au bout, malgré ma situation de retraité.

Je suis très reconnaissant auprès de mes collègues informaticiens de Philips Composants (ex Radiotechnique Compelec - R.T.C.) à Caen, mon premier employeur en 1982, pour m'avoir donné envie de ne pas rester dans l'industrie et de chercher un emploi dans la recherche.

Je remercie aussi mes collègues du S.E.P.T. pour m'avoir offert mon premier emploi dans un centre de recherche. Je pense tout particulièrement à Jean-Marc Deshayes, avec qui nous avons démarré une aventure de recherche qui restera fondatrice tout au long de mon parcours ultérieur, mais aussi à Philippe Maurice, l'un de mes supérieurs très compréhensif et Yves Londechamp qui a eu l'audace de miser sur de jeunes diplômés dans une confiance remarquable et indéfectible.

Je n'oublie pas tous les enseignants qui m'ont donné envie de poursuivre en informatique, à commencer par Bernard Victorri, dont j'ai suivi, en auditeur libre, son cours en I.A. à l'Université de Caen. À cette époque j'étais très très loin de penser que quelques années plus tard je serais Professeur responsable de l'option I.A. au DEA «Intelligence Artificielle et Algorithmique» - IAA, et en charge d'un cours que j'ai créé sur les «Systèmes

Multi-Agents» - SMA, à cette même Université de Caen. Jean-Claude Simon, enseignant dans le DEA «Traitement algorithmique de l'information» à l'Université Paris VI - Jussieu, aura été influant sur mon parcours ultérieur. Une bonne partie de son cours (en 1982) consistait à faire venir des chercheurs de tous les labos de France pour nous présenter leur travail. On ne pouvait pas imaginer meilleure ouverture sur la recherche.

Je remercie aussi Jean-Pierre Müller et toute l'équipe des rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels de Rochebrune, pour cette initiative scientifique féconde, dont la richesse repose sur l'interdisciplinarité. Ce rendez-vous aura marqué ma carrière avec trois étapes décisives: d'abord en y rencontrant Fabrice Harrouet et Roger Cozien, qui présentaient *oRis*, puis en soumettant l'article sur la singularité (cf. le chapitre II), enfin en soumettant l'article sur l'épaisseur du temps (cf. le chapitre VIII).

Merci à:

- Bernard Victorri, Jean-Marc Fouet, Martial Vivet, Sylvie Simon, Jean-Bernard Stéfani, Paul Bourgine, Philippe Maurice, Christophe Sibertin-Blanc, Guy Bernard, Patrice Enjalbert, Anne Nicolle, Brahim Chaib-draa, Hervé Guyennet, Georges Le Noane (membres du jury de ma thèse et de mon HDR);
- Yves Demazeau qui a cru en mes travaux et tout spécialement cette idée un peu folle d'utiliser les nuées d'oiseaux pour imaginer un système de stockage de documents; idée qui a pu se développer grâce à l'opiniâtreté d'Hugo Pommier et de Benoît Romito;
- Bruno Dillenseger, Yvan Peter, Eric Malville, Luigi Lancieri, Nicolas Saillard, Salah El Falou, Van Tuan LE, Hugo Pommier, Sébastien Picant, Benoît Romito (étudiants ayant soutenu leur thèse sous ma direction), pour m'avoir fait confiance;
- tous les étudiants que j'ai encadré dans leur stage de DEA, de DESS ou de fin d'études à l'ENSICAEN;
- tous les étudiants du département informatique de l'I.U.T. de Caen qui ont fait un stage de fin d'étude sur mes thématiques de recherche et ceux qui ont goûté à la recherche avec moi, et d'autres collègues, et qui sont devenus chercheurs eux-mêmes.

Enfin je dois remercier Anita Rigot qui a été le catalyseur de la réalisation de ce testament. Je portais l'idée en moi, mais elle a su me convaincre de me lancer dans ce projet un peu fou en fin de carrière.

Je suis entré en recherche par une thèse, j'en sors par ce testament qui pourrait être défendu comme une nouvelle thèse.

L'Estaca, vendredi 20 octobre 2023.

«**SINGULARITÉ** ou l'émergence d'un paradoxe fécond, Testament de recherche 2019-2023». La charte graphique a été très largement inspirée, avec l'accord de son auteur que je remercie, de la thèse de Valentin Bourdon intitulée «Les formes architecturales du Commun», école polytechnique fédérale de Lausanne, soutenue le 29 septembre 2020. Droits Réservés. francois@bourdon.info / 06 72 10 24 65. (France).

Il m'aura fallu quatre années pour mener à bien ce projet de Testament de recherche, dont les deux dernières sous le régime de la retraite avec un statut de chercheur associé au GREYC. Deux jalons principaux (mon doctorat en 1992 et mon HDR en 1998) me permirent de baliser plusieurs axes de recherche situés dans le champ de l'informatique, plus précisément dans celui de l'intelligence artificielle distribuée, appelée communément les «systèmes multi-agents», et encore plus précisément sur l'émergence dans les comportements collectifs.

Le problème étudié ici est celui du placement optimum de ressources en vue du traitement réparti et décentralisé, d'informations dans un environnement ouvert, incertain et à grande échelle, tel que l'apparaît aujourd'hui Internet. L'approche consiste à proposer des algorithmes basés sur un contrôle entièrement décentralisé.

Ces travaux peuvent se résumer en trois points. Le premier est une méthode de résolution (MVB) de l'allocation dynamique de ressources partagées. Grâce à cette approche heuristique, fondée sur une étude approfondie du problème, des résolutions fines et non coûteuses en temps de calcul, ont été obtenues, basées sur l'exploration sans apprentissage de l'espace des solutions via un processus auto-organisé. Le deuxième réside dans des perspectives à cette approche, dans plusieurs domaines comme celui de la représentation des connaissances, ou encore celui du placement crypté de nuées de serveurs de données.

Le troisième et dernier point réunit toutes ces perspectives dans un principe «philosophique» qui fait l'éloge de la lenteur. Ce principe peut être considéré comme l'un des résultats le plus intéressant à nos yeux de tous ces travaux, dans la mesure où il a une portée qui les dépasse largement, et n'était absolument pas recherché au départ (sérendipité). Il constitue le fil rouge de ma quête incarnée par mes derniers résultats obtenus dans la projection de MVB vers un monde distribué et ouvert non simulé (Internet), mais aussi, et de façon beaucoup plus surprenante dans mon expérience artistique avec l'encre.

A plus long terme, ce travail me permet de proposer une vision quantique de la représentation des connaissances en introduisant, toujours dans MVB, le modèle de *dérive des connaissances*, défini dans ma thèse. Les phénomènes émergents, dont les nuées d'oiseaux en sont l'un des exemples les plus parlants, restent et resteront pour moi d'un immense intérêt. Ils sont source d'imagination, d'émerveillement et titillent, comme rien d'autre, mon appétit de curiosité. Je leur en sais gré.