A Novel Algorithm for Finding Maximum Common Ordered Subgraph

Nicola Yanev, Rumen Andonov

▶ **To cite this version:**

Nicola Yanev, Rumen Andonov. A Novel Algorithm for Finding Maximum Common Ordered Subgraph. [Research Report] RR-6287, INRIA. 2007, pp.18. inria-00171780v2

HAL Id: inria-00171780
https://inria.hal.science/inria-00171780v2

Submitted on 14 Sep 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Novel Algorithm for Finding Maximum Common Ordered Subgraph

Nicola Yanev — Rumen Andonov

N° 6287

Septembre 2007

Thème BIO

ISSN 0249-6399 ISRN INRIA/RR--6287--FR+ENG

A Novel Algorithm for Finding Maximum Common Ordered Subgraph

Nicola Yanev * , Rumen Andonov †

Thème BIO — Systèmes biologiques
Projets SYMBIOSE

Rapport de recherche n° 6287 — Septembre 2007 —15 pages

Abstract: In this paper, we study the following problem: given are adjacency matrices of two simple graphs. Find two principal matrices (though they are vectors) having the maximum inner product. When used for computing the similarity of two protein structures this problem is called contact map overlap and for the later, we give an exact B&B algorithm with bounds computed by solving Lagrangian relaxation of the problem. The efficiency of the approach is demonstrated on a popular benchmark set of instances together with a comparison with the best existing algorithm.

Key-words: combinatorial optimization, graph algorithms, proteins structures comparison

Supported by ANR grant Calcul Intensif projet PROTEUS (ANR-06-CIS6-008) and by Hubert Curien French-Bulgarian partnership "RILA 2006" N^0 15071XFT

* University of Sofia, Bulgaria
† IRISA and University of Rennes 1, Campus de Beaulieu, 35042 Rennes, France

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Un nouvel algorithme pour la récherche du plus grand sous-graphe commun ordonné

Résumé : Dans cet article, nous étudions le problème suivant : étant donné deux matrices d'adjacences de deux graphes simples, trouver deux matrices principales (en faite, deux vecteurs) ayant le plus grand produit scalaire. Quand il est utilisé pour calculer la similarité de deux structures de protéines, ce problème est appelé « Contact Map Overlap » (CMO), et par la suite, nous montrons un algorithme de branch and bound exacte, dont les bornes sont calculées en résolvant la relaxation lagrangienne de ce problème. L'efficacité de cette approche est démontrée sur un jeu de test d'instances réelles, en comparaison avec le meilleur algorithme existant.

Mots-clés : optimisation combinatoire, algorithmique de graphes, comparaison de structures protéiques

1 Introduction

It is a fundamental axiom of biology that the 3-dimensional (3D) structure of a protein has a crucial influence on its function- two proteins that are similar in their 3D structure will likely have similar functions. Comparing two protein structures for similarity is therefor a crucial task and has been extensively investigated [8].
Since it is not clear what quantitative measure to use for comparing protein structures, a multitude of measures have been proposed. Each measure aims in capturing the intuitive notion of similarity. We study the *contact-map-overlap* (CMO) measure, first proposed in [9]. This measure has been found to be very useful for measuring protein similarity - it is robust, takes partial matching into account, translation invariant and captures the intuitive notion of similarity very well for details. Thus the problem of designing efficient algorithms that guarantee the CMO quality is an important one that has eluded researchers so far.

Here, we present an algorithm for exact solving the CMO problem (the formal definition is given below). The CMO is just one of the scoring schemes used for comparison of protein structures. The protein's primary sequence is usually though-of as composed of residues. Under specific physiological conditions, the linear arrangement of residues will fold and adopt a complex three dimensional shape, called native state (or tertiary structure) of the protein. In its native state, residues that are far away along the linear arrangement may come into proximity in three dimensional space. The proximity relation between residues in a protein is captured by a mathematical construct called a contact map. Formally, a map is specified by a $0-1$ symmetric $n \times n$ matrix C whose 1-elements correspond to pairs of amino acids on 3D contact, i.e. $c_{ij} = 1$ if the Euclidean distance of two heavy atoms (or the minimum distance between any two atoms belonging to those residues) from the i-th and the j-th amino acid of a protein is smaller than a given threshold in the protein native fold. In the pairwise comparison one tries to evaluate the similarity in the 3D- folds of two proteins by determining the maximum overlap (also called alignment) of contacts map. This can be formulated in the following manner: given are adjacency matrices of two simple graphs. Find two principal matrices[1] (though they are vectors) having the maximum inner product.

Our interest in the comparison of the $3D$ structures of protein molecules based on such maximal common sub-graph detection is provoked by the apparent similarity of the resulting optimization problem with the one derived by another challenging problem- the protein folding, approached by the protein threading technique. For the later in [1] we have presented a methodology, based on s.c. non-crossing matching in bipartite graphs, culminated in highly efficient algorithms for solving the PTP by using the Lagrangian duality [2, 3, 4]. In the same time (independently) in [5] a Lagrangian approach have been reported successful for the CMO problem. Please refer to this paper for the history of this problem, the various techniques for solving it, and at the end for the triumph of the algorithm proposed there. So the challenge is : could one create a competitive algorithm based on the above-mentioned PTP platform ? Below, we concentrate on the description of such an algorithm and answer affirmatively to this question. The counterpart of the CMO problem in the graph theory is the well known maximum common subgraph problem (MCS) [10]. The bad news for the later is its APX-hardness (see *A compendium of NP optimization problems* available at `http://www.nada.kth.se/~viggo/problemlist/`). The only difference between

[1] a principal matrix is a sub-matrix of a squared matrix obtained by deleting k rows and the same k columns

the above defined CMO and MCS is that the isomorphism used for the MCS is not restricted to the non-crossing matching only. Nevertheless the CMO is also known [6] to be NP-hard. Some authors [7] use the adjectives sequential, if in doing protein structure alignment the sequential order (the vertices of the graphs are ordered by a linear sequence and the bijection is order-preserving) is enforced, and non-sequential (equivalent to MCS) in the other case.

2 The mathematical model

We are going to present the CMO problem as a matching problem in a bipartite graph, which in turn will be posed as a longest augmented path problem in a structured graph. Toward this end we need to introduce few notations as follows. The contacts maps of two proteins P1 and P2 are given by graphs $G_m = (V_m, E_m)$ with $V_m = \{1, 2, \ldots, n_m\}$ for $m = 1, 2$. The vertices V_m are better seen as ordered points on a line and correspond to the residues of the proteins. The arcs (i, j) correspond to the contacts. The right and left neighboring of node i are elements of the sets $\delta_m^+(i) = \{j | j > i, (i, j) \in E_m\}$, $\delta_m^-(i) = \{j | j < i, (j, i) \in E_m\}$. Let $i \in V_1$ be matched with $k \in V_2$ and $j \in V_1$ be matched with $l \in V_2$. We will call a matching *non-crossing*, if $i < j$ implies $k < l$. A feasible alignment of two proteins P_1 and P_2 is given by a non-crossing matching in the complete bipartite graph B with a vertex set $V_1 \cup V_2$.

Let the weight w_{ikjl} of the matching couple $(i,k)(j,l)$ be set as follows

$$w_{ikjl} = \begin{cases} 1 & \text{if } (i,j) \in E_1 \text{ and } (k,l) \in E_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For a given non-crossing matching M in B we define its weight $w(M)$ as a sum over all couples of edges in M. The CMO problem consists then in maximizing $w(M)$, where M belongs to the set of all non-crossing matching in B.

In [1, 2, 3, 4] we have already dealt with non-crossing matching and we have proposed a network flow presentation of similar one-to-one mappings (in fact the mapping there was many-to-one). The adaptation of this approach to CMO is as follows: The edges of the bipartite graph B are mapped to the points of $n_1 \times n_2$ rectangular grid $B' = (V', E')$ according to: point - $(i,k) \in V' \longleftrightarrow$ edge - (i,k) in B.

Definition. The **feasible path** is an arbitrary sequence $(i_1, k_1), (i_2, k_2), \ldots, (i_t, k_t)$ of points in B' such that $i_j < i_{j+1}$ and $k_j < k_{j+1}$ for $j = 1, 2, \ldots, t-1$.

The correspondence feasible path $\longleftrightarrow$ non-crossing matching is obvious. This way the problems on non-crossing matching are converted to problems on feasible paths. We also add arcs $(i,k) \rightarrow (j,l) \in E'$ iff $w_{ikjl} = 1$. In B', solving CMO corresponds to finding the densest (in terms of arcs) subgraph of B' whose node set is a feasible path (see for illustration Fig. 1).

To each node $(i,k) \in V'$ we associate now a $0/1$ variable x_{ik}, and to each arc $(i,k) \rightarrow (j,l) \in E'$, a $0/1$ variable y_{ikjl}. Denote by X the set of feasible paths. The problem can now be stated as follows (see Fig. 2 a) for illustration)

$$v(CMO) = \max \sum_{(ik)(jl) \in E'} y_{ikjl} \tag{2}$$

subject to

$$x_{ik} \geq \sum_{l \in \delta_2^+(k)} y_{ikjl}, \quad j \in \delta_1^+(i) \qquad \begin{array}{l} i = 1, 2, \ldots, n1-1, \\ k = 1, 2, \ldots, n2-1 \end{array} \tag{3}$$

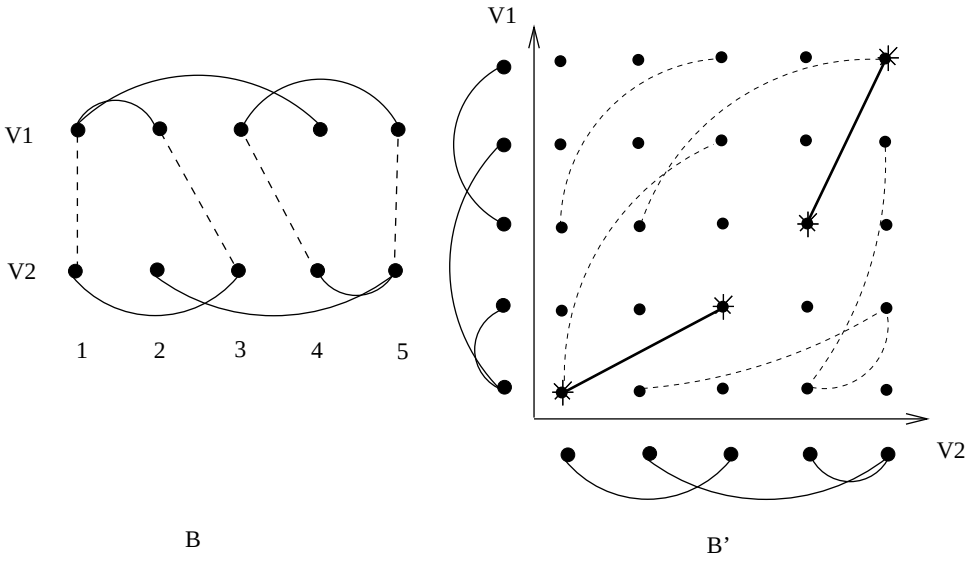

Figure 1: Left: Vertex 1 from V1 is matched with vertex 1 from V2 and 2 is matched with 3: matching couple $(1,1)(2,3)$. Other matching couples are $(3,4)(5,5)$. This defines a feasible matching $M = \{(1,1)(2,3),(3,4)(5,5)\}$ with weight $w(M) = 2$. Right: The same matching is visualized in graph B'.

$$x_{ik} \geq \sum_{l \in \delta_2^-(k)} y_{jlik}, \quad j \in \delta_1^-(i) \qquad \begin{array}{l} i = 2,3,\ldots,n1, \\ k = 2,3,\ldots,n2 \end{array} \tag{4}$$

$$x_{ik} \geq \sum_{j \in \delta_1^+(i)} y_{ikjl}, \quad l \in \delta_2^+(k) \qquad \begin{array}{l} i = 1,2,\ldots,n1-1, \\ k = 1,2,\ldots,n2-1 \end{array} \tag{5}$$

$$x_{ik} \geq \sum_{j \in \delta_1^-(i)} y_{jlik}, \quad l \in \delta_2^-(k) \qquad \begin{array}{l} i = 2,3,\ldots,n1, \\ k = 2,3,\ldots,n2. \end{array} \tag{6}$$

$$x \in X \tag{7}$$

Actually, we know how to represent X with linear constraints. Recalling the definition of feasible path, (7) is equivalent to

$$\sum_{l=1}^{k} x_{il} + \sum_{j=1}^{i-1} x_{jk} \leq 1, \quad i = 1,2,\ldots,n1, \quad k = 1,2,\ldots,n2. \tag{8}$$

We recall that from the definition of the feasible paths in B' (non-crossing matching in B) the j-th residue from $P1$ could be matched with at most one residue from $P2$ and vice-versa. This explains the sums into right hand side of (3) and (5) – for arcs having their tails at vertex (i,k); and (4) and (6)– for arcs heading to (i,k). Any $(i,k)(j,l)$ arc can be activated ($y_{ikjl} = 1$) iff $x_{ik} = 1$ and $x_{jl} = 1$ and in this case the respective constraints are active because of the objective function.

A tighter description of the polytop defined by (3)–(6) and $0 \leq x_{ik} \leq 1$, $0 \leq y_{ikjl}$ could be obtained by lifting the constraints (4) and (6) as it is shown in Fig. 2 b). The points shown are just the predecessors of (i,k) in graph B' and they form a grid of $\delta_1^-(i)$ rows and $\delta_2^-(k)$ columns. Let $i_1, i_2, \ldots, i_s$ be all the vertices in $\delta_1^-(i)$ ordered according the numbering of the vertices in V_1 and likewise $k_1, k_2, \ldots, k_t$ in $\delta_2^-(k)$. Then the vertices in the l-th column $(i_1,k_l),(i_2,k_l),\ldots(i_s,k_l)$ correspond to pairwise crossing matching and at most one of them could be chosen in any feasible solution $x \in X$ (see (6)). This "all crossing" property will stay even if we add to this set the following two sets: $(i_1,k_1),(i_1,k_2),\ldots,(i_1,k_{l-1})$ and $(i_s,k_{l+1}),(i_s,k_{l+2},\ldots,(i_s,k_t)$. Denote by $col_{ik}(l)$ the union of these three sets and analogously by $row_{ik}(j)$ the corresponding union for the j-th row of the grid. When the grid is one column/row only the set $row_{ik}(j)/col_{ik}(l)$ is empty.

Now a tighter LP relaxation of (3)–(6) is obtained by changing (4) with

$$x_{ik} \geq \sum_{(r,s)\in row_{ik}(j)} y_{rsik}, \quad j \in \delta_1^-(i) \qquad \begin{matrix} i = 2,3,\ldots,n1, \\ k = 2,3,\ldots,n2 \end{matrix} \tag{9}$$

and (6) with

$$x_{ik} \geq \sum_{(r,s)\in col_{ik}(l)} y_{rsik}, \quad l \in \delta_2^-(k) \qquad \begin{matrix} i = 2,3,\ldots,n1, \\ k = 2,3,\ldots,n2. \end{matrix} \tag{10}$$

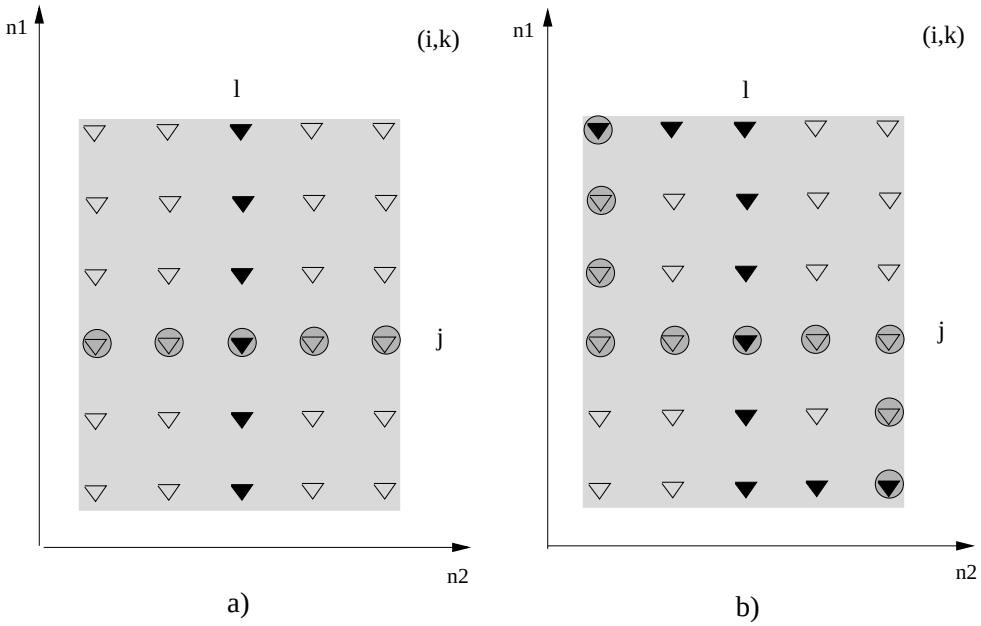

Figure 2: The shadowed area represents the set of vertices in V' which are tails for the arcs heading to (i,k). In a): ▼ corresponds to the indices of y_{jlik} in (6) for l fixed. ◯ corresponds to the indices of y_{jlik} in (4) for j fixed. In b): ▼ corresponds to the indices of y_{jlik} in (10) for l fixed (the set $col_{ik}(l)$). ◯ corresponds to the indices of y_{jlik} in (9) for j fixed (the set $row_{ik}(j)$).

Remark: Since we are going to apply the Lagrangian technique there is no need neither for an explicit description of the set X neither for lifting the constraints (3) (5).

3 Lagrangian relaxation approach

Here, we show how the Lagrangian relaxation of constraints (9) and (10) leads to an efficiently solvable problem, yielding upper and lower bounds that are generally better than those found by the best known exact algorithm [5].

Let $\lambda^h_{ikj} \geq 0$ (respectively $\lambda^v_{ikj} \geq 0$) be a Lagrangian multiplier assigned to each constraint (9) (respectively (10)). By adding the slacks of these constraints to the objective function with weights λ, we obtain the Lagrangian relaxation of the CMO problem

$$\begin{aligned} LR(\lambda) = \quad & \max \sum_{i,k,j \in \delta_1^-(i)} \lambda^h_{ikj}(x_{ik} - \sum_{(r,s) \in row_{ik}(j)} y_{rsik}) \\ & + \sum_{i,k,l \in \delta_2^-(k)} \lambda^v_{ikl}(x_{ik} - \sum_{(r,s) \in col_{ik}(l)} y_{rsik}) + \sum_{(ik)(jl) \in E_{B'}} y_{ikjl} \end{aligned} \tag{11}$$

subject to $x \in X$, (3), (5) and $y \geq 0$.

Proposition 1 *$LR(\lambda)$ can be solved in $O(|V'| + |E'|)$ time.*

Proof:

For each $(i,k) \in V'$, if $x_{ik} = 1$ then the optimal choice y_{ikjl} amounts to solving the following : The heads of all arcs in E' outgoing from (i,k) form a $|\delta^+(i)| \times |\delta^+(k)|$ table. To each point (j,l) in this table, we assign the profit $\max\{0, c_{ikjl}(\lambda)\}$, where $c_{ikjl}(\lambda)$ is the coefficient of y_{ikjl} in (11). Each vertex in this table is a head of an arc outgoing from (i,k).Then the subproblem we need to solve consists in finding a subset of these arcs having a maximal sum $c_{ik}(\lambda)$ of profits(the arcs of negative weight are excluded as a candidates for the optimal solution) and such that their heads lay on a feasible path. This could be done by a dynamic programming approach in $O(|\delta^+(i)||\delta^+(k)|)$ time. Once profits $c_{ik}(\lambda)$ have been computed for all (i,k) we can find the optimal solution to $LR(\lambda)$ by using the same DP algorithm but this time on the table of $n1 \times n2$ points with profits for (i,k)-th one given by

$$c_{ik}(\lambda) + \sum_{j \in \delta_1^-(i)} \lambda^h_{ikj} + \sum_{l \in \delta_2^-(k)} \lambda^v_{ikl}. \tag{12}$$

where the last two terms are the coefficients of x_{ik} in (11).

Remark: The inclusion $x \in X$ is explicitly incorporated in the DP algorithm.

3.1 The algorithm

In order to find the tightest upper bound on $v(CMO)$ (or eventually to solve the problem), we need to solve in the dual space of the Lagrangian multipliers $LD = \min_{\lambda \geq 0} LR(\lambda)$, whereas $LR(\lambda)$ is a problem in x,y. A number of methods have been proposed to solve Lagrangian duals: subgradient method, dual ascent methods,constraint generation method, column generation, bundel methods,augmented Lagrangian methods, etc. Here, we choose the subgradient method. It is an iterative method in which at iteration t, given the current multiplier vector λ^t, a step is taken along a subgradient of $LR(\lambda)$, then if necessary, the resulting point is projected onto the nonnegative orthant. It is

well known that practical convergence of the subgradient method is unpredictable. For some problems, convergence is quick and fairly reliable, while other problems tend to produce erratic behavior of the multiplier sequence, or the Lagrangian value, or both. In a "good" case, one usually observe a saw-tooth pattern in the Lagrangian value for the first iterations, followed by a roughly monotonic improvement and asymptotic convergence to a value that is hopefully the optimal Lagrangian bound. The computational runs on a reach set of real-life instances confirm a "good" case belonging of our approach at some expense in the speed of the convergence.

In our realization, the update scheme for λ_{ikj} is $\lambda_{ikj}^{t+1} = \max\{0, \lambda_{ikj}^{t} - \Theta^t g_{ikj}^t\}$ [2], where $g_{ikj}^t = \bar{x}_k - \sum \bar{y}_{lik}$ (see (9) and (10) for the sum definition) is the sub-gradient component $(0, 1, \text{or} -1)$, calculated on the optimal solution $\bar{x}$, $\bar{y}$ of $LR(\lambda^t)$. The step size Θ^t is $\Theta^t = \frac{\alpha(LR(\lambda^t) - Z_{lb})}{\sum (g_{ikj}^t)^2 + \sum (g_{ikl}^t)^2}$ where Z_{lb} is a known lower bound for the CMO problem and α is an input parameter. Into this approach the x-components of $LR(\lambda^t)$ solution provides a feasible solution to CMO and thus a lower bound also. The best one (incumbent) so far obtained is used for fathoming the nodes whose upper bound falls below the incumbent and also in section 4 for reporting the final gap. If $LD \leq v(CMO)$ then the problem is solved. If $LD > v(CMO)$ holds, in order to obtain the optimal solution, one could pass to a branch&bound algorithm suitably tailored for such an upper bounds generator.

From among various possible nodes splitting rules, the one shown in Fig. 3 gives quite satisfactory results (see section 4). Formally, let the current node be a subproblem of CMO defined over the vertices of V' falling in the interval $[lc(k), uc(k)]$ for $k = 1, n_2$ (in Fig. 3 these are the points in-between two broken lines (the white area). Let $(rowbest, colbest)$ be the $\arg\max\min(S_u(i,k), S_d(i,k))$, where $S_d(i,k) = \sum_{j \leq k} \max(uc(j) - i, 0)$ and $S_u(i,k) = \sum_{j \geq k} \max(i - lc(j), 0)$. Now, the two descendants of the current node are obtained by discarding from its feasible set the vertices in $S_d(rowbest, colbest)$ and $S_u(rowbest, colbest)$ respectively. The goal of this strategy is twofold: to create descendants that are balanced in sense of feasible set size and to reduce maximally the parent node's feasible set.

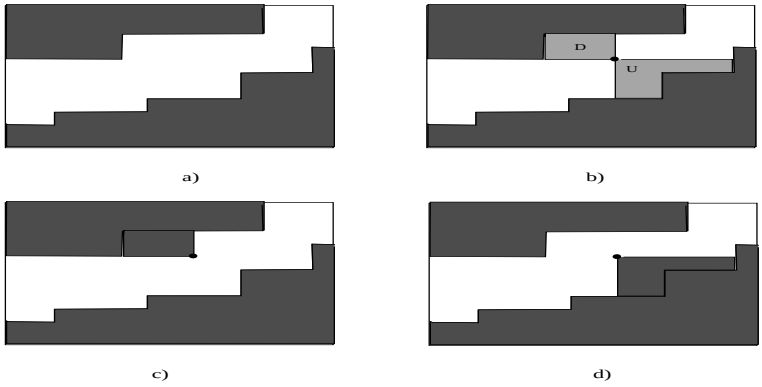

Figure 3: Illustration of the B&B splitting strategy. a) the white area in-between the two broken lines represents the current node feasible set; b) This set is split by $(rowbest, colbest)$, D corresponds to the set $S_d(rowbest, colbest)$, while U corresponds to the set $S_u(rowbest, colbest)$; c) and d) are the two descendants of the node a).

[2] analogously for λ_{ikl}

In addition, the following heuristics happened to be very effective during the traverse of the B&B tree nodes. Once the lower and the upper bound are found at the root node, an attempt to improve the lower bound is realized as follows.

Let $(i_{k_1},k_1),(i_{k_2},k_2),\ldots,(i_{k_s},k_s)$ be an arbitrary feasible path which activates certain number of arcs (recall that each iteration in the sub-gradient optimization phase generates such path and lower bound as well).

Then for a given strip size *sz* (an input parameter set by default to 4), the matchings in the original CMO are restricted to fall in a neighborhood of this path, allowing x_{ik} to be non zero only for

$$\max\{1,i_j-sz\}\leq i\leq \min\{n1,i_j+sz\}, j=k_1,k_2,\ldots,k_s.$$

The Lagrangian dual of this subproblem is solved and a better lower bound is possibly sought. If the bound improves the incumbent,the same procedure is repeated by changing the strip alongside the new feasible solution.

Finally, the main steps of the B&B algorithm are as follows:
Initialization: Set L={original CMO problem, i.e. no restrictions on the feasible paths}.
Problem selection and relaxation: Select and delete the problem P^i from L having the biggest upper bound. Solve the Lagrangian dual of P^i. (Here a repetitive call to a heuristics is included after each improvement on the lower bound).
Fathoming and Pruning: Follow the classical rules.
Partitioning : Create two descendants of P^i using $(rowbest,colbest)$. Add these descendants to L.
Termination : if $L=\emptyset$, the solution (x^*,y^*) which yielded the incumbent objective value is optimal.

4 Computational results

The numerical results presented in this section were obtained on a cluster of 12 AMD Opteron(TM) CPU 2.4 GHz, 4 Gb Ram, RedHat 9 Linux, connected by a 1 Gb Ethernet network. The algorithm was implemented in C. To test its performance we used a set of large proteins suggested by Jeffrey Skolnick that was used in various recent papers related to protein structure comparison [5, 11]. This set contains 33 proteins with a total of 40 domains classified by SCOP into five families (see Table 1) [3]. Below we compare the performance of our approach with the previously known exact algorithm [5]. Note that both approaches use diverse (but Lagrangian type) relaxations. Our algorithm will be called `a_purva`[4], while the other Lagrangian algorithm will be denoted here by (`LR`)[5]. The Skolnick set requires aligning 780 pairs of proteins. Those are medium size proteins, the number of their residues varies from 95 (2b3iA) to 252 (1aw2A). The maximum number of contacts is 593 (1btmA). We bounded the execution time to 1800 seconds for both algorithms. `a_purva` succeeded to solve 171 couples for the given period of time, while `LR` solved only 157 couples. Figure 4 illustrates `LR`/`a_purva` time ratio as a function of solved instances. It is easily seen that

[3] Caprara et al. [5] mention only four families. This wrong classification is also accepted in other studies [11]. The families are in fact five as shown in Table 1. According to SCOP classification the protein 1arn1 does not belong to the first family as indicated in [5]. Note that this corroborates the results obtained in [5] but the authors considered it as a mistake.

[4] Apurva (Sanskrit) = not having existed before, unknown, wonderful, ...

[5] The code of LR, as well as the contact map graphs for the Skolnick set, were kindly provided to us by Giuseppe Lancia.

	Fold	Family	Proteins
1	Flavodoxin-like	CheY-related	1b00, 1dbw, 1nat, 1ntr, 1qmp(A,B,C,D), 3chy, 4tmy(A,B)
2	Cupredoxin-like	Plastocyanin/ azurin-like	1baw, 1byo(A,B), 1kdi, 1nin, 1pla 2b3i, 2pcy, 2plt
3	TIM beta/alpha-barrel	Triosephosphate isomerase (TIM)	1amk, 1aw2, 1b9b, 1btm, 1hti 1tmh, 1tre, 1tri, 1ydv, 3ypi, 8tim
4	Ferritin-like	Ferritin	1b71, 1bcf, 1dps, 1hfa, 1ier, 1rcd
5	Microbial ribonucleases	Fungal ribonucleases	1rn1(A,B,C)

Table 1: The Skolnick set

a_purva is significantly faster than LR (up to several hundred times in the majority of cases). Table 2 in the Appendix contains more details concerning a subset of 164 pairs of proteins. We observed that this set is a very interesting one. It is characterized by the following properties: a) in all but the 6 last instances the a_purva running time is, less than 10 seconds; b) in all instances the relative gap[6] at the root of the B&B is smaller than 4, while in all other instances this gap is much larger : greater than 18 even for the couples we succeeded to solve for less than 1800 sec: c) this set contains all instances such that both proteins belong to the same family according SCOP classification. In other words, each pair such that both proteins belong to the same family is an easily solvable instance for a_purva and this feature can be successfully used as a discriminator (at least for the Skolnick set). In fact, by virtue of this relation (similar structure-less computational time and vice versa), we were able to correctly classify this 40 items set in 2000 seconds overall running time an all 780 instances.

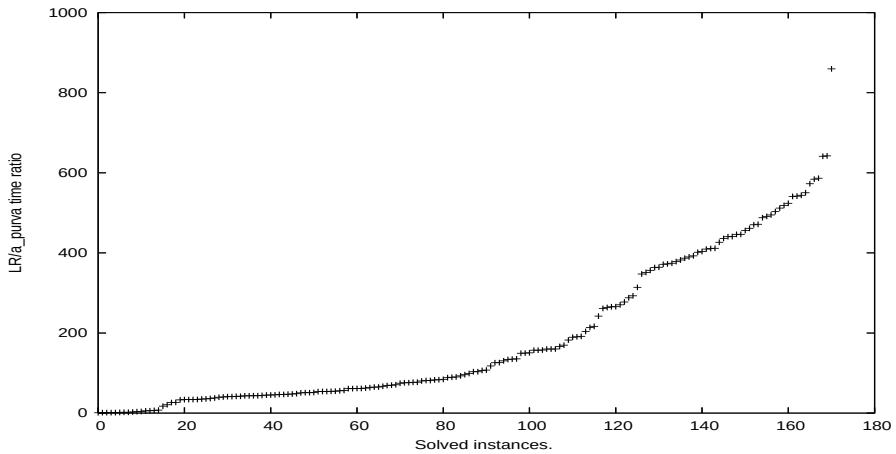

Figure 4: $\frac{\text{LR time}}{\text{a\_purva time}}$ ratio as a function of solved instances

Our next observation (see Figures 5 and 6) concerns the quality of gaps obtained by both algorithms on the set of unsolved instances. Remember that when a Lagrangian algorithm stops because of time limit (1800 sec. in our case) it provides two bounds:

[6] We define the relative gap as $100 \times \frac{UB-LB}{UB}$.

one upper (UB), and one lower (LB). Providing these bounds is a real advantage of a B&B type algorithm compared to any meta-heuristics. These values can be used as a measure for how far is the optimization process from finding the exact optimum. The value UB-LB is usually called absolute gap. Any one of the 609 points (x,y) in Figure 5) presents the absolute gap for `a_purva` (x coordinate) and for `LR` (y coordinate) algorithm. All points are above the $y = x$ line (i.e. the absolute gap for `a_purva` is always smaller than the absolute gap for `LR`). On the other hand the entire figure is very asymmetric in a profit of our algorithm since its maximal absolute gap is 33, while it is 183 for `LR`.

We afterwards similarly compared lower and upper bounds separately. This is illustrated in Fig. 6. Any point denoted by $\circ$ has the lower bound computed by `a_purva(LR)` as x (y) coordinate, while any point denoted by $\times$ has the upper bound computed by `a_purva(LR)` as x (y) coordinate. We observe that in a large majority the points $\circ$ are below the $y = x$ line while the points $\times$ are above this line. This shows that usually the lowers bounds found by `a_purva` are higher, while its upper bounds are all smaller and it is clear that `a_purva` significantly outperforms `LR` on quality of its bounds.

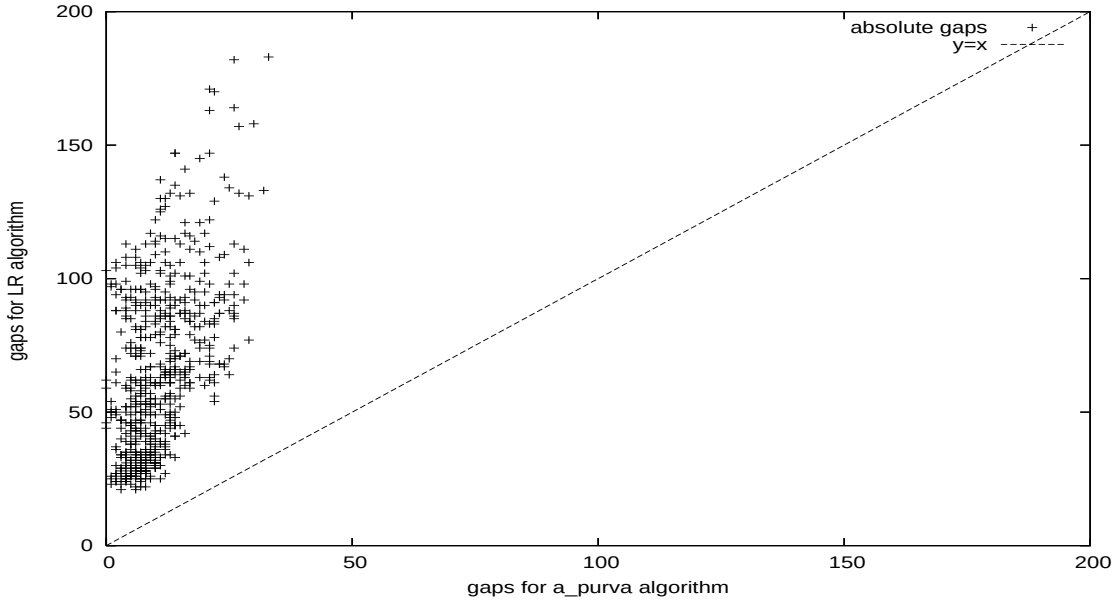

Figure 5: Comparing absolute gaps on the set of unsolved instances. The gaps computed by a_purva are significantly smaller.

4.1 Conclusion

In this paper, we give efficient exact B&B algorithm for contact map overlap problem . The bounds are found by using Lagrangian relaxation and the dual problem is solved by sub-gradient approach. The efficiency of the algorithm is demonstrated on a benchmark set of 780 instances and the dominance over the existing algorithms is total. When the

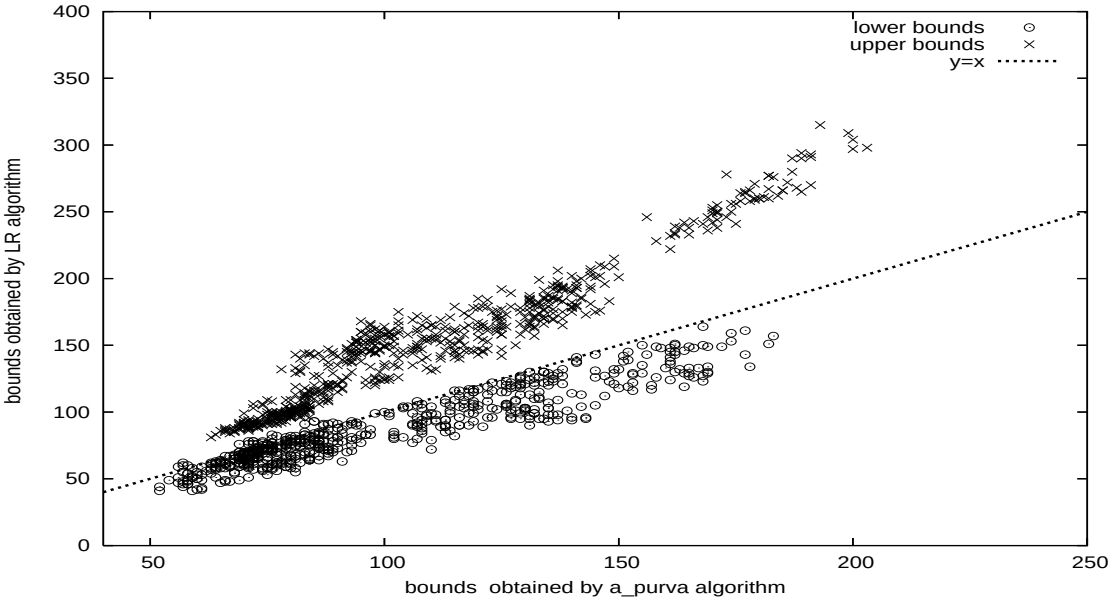

Figure 6: Comparing the quality of lower and upper bounds on the set of unsolved instances. a_purva clearly outperforms LR on the quality of its bounds.

algorithm is used for classification purposes (and this was the primary goal) the average time for correctly classifying two proteins of the same class is 0.6 seconds.

References

[1] R. Andonov, S. Balev, and N. Yanev. Protein threading: From mathematical models to parallel implementations. *INFORMS Journal on Computing*, 16(4), 2004.

[2] S. Balev. Solving the protein threading problem by lagrangian relaxation. In *Proceedings of WABI 2004: 4th Workshop on Algorithms in Bioinformatics*, LNCS/LNBI. Springer-Verlag, 2004.

[3] P. Veber, N. Yanev, R. Andonov, V. Poirriez. Optimal protein threading by cost-splitting *Lecture Notes in Bioninformatics*, 3692, pp.365-375, 2005

[4] N. Yanev, P. Veber, R. Andonov and S. Balev. Lagrangian approaches to a class of matching problems *INRIA-00090635-v2*, 2006 and in *Journal of computational and applied mathematics*, 2007 (to appear)

[5] A. Caprara, R. Carr, S. Israil, G. Lancia and B. Walenz, 1001 Optimal PDB Structure Alignments: Integer Programming Methods for Finding the Maximum Contact Map Overlap *Journal of Computational Biology*, 11(1), 2004, pp. 27-52

[6] D. Goldman, C.H. Papadimitriu, and S. Istrail. Algorithmic aspects of protein structure similarity *FOCS 99: Proceedings of the 40th annual symposium on foundations of computer science* IEEE Computer Society,1999

[7] J. Xu, F. Jiao, B. Berger. A parametrized Algorithm for Protein Structer Alignment *RECOMB 2006, Lecture Notes in Bioninformatics*, 3909,pp. 488-499,2006

[8] I. Halperin, B. Ma, H. Wolfson, et al. Principles of docking: An overview of search algorithms and a guide to scoring functions *Proteins Struct. Funct. Genet.*, 47, 409-443,2002

[9] D. Goldman, S. Israil, C. Papadimitriu. Algorithmic aspects of protein structure similarity *IEEE Symp. Found. Comput. Sci.* 512-522,1999

[10] M. Garey, D. Johnson. Computers and Inractability: A Guide to the Thoery of NP-completness *Freeman and company*, New york, 1979

[11] D. Pelta, N. Krasnogor, C. Bousono-Calzon, J. L. Verdegay, J. Hirst, E. Burke. A fuzzy sets based generalization of contact maps for the overlap of protein structures *Journal of Fuzzy Sets and Systems*, 152(2):103-123, 2005.

APPENDIX

F	Proteins Name	CMO	Time LR	Time a_pr	Proteins Name	CMO	Time LR	Time a_pr
1	1b00A 1dbwA	149	192.00	1.2	1ntr_ 1qmpA	119	545.94	7.18
1	1b00A 1nat_	145	166.98	1.11	1ntr_ 1qmpB	115	454.01	4.23
1	1b00A 1ntr_	118	565.47	3.59	1ntr_ 1qmpC	116	610.93	6.56
1	1b00A 1qmpA	143	198.72	1.33	1ntr_ 1qmpD	118	522.53	4.44
1	1b00A 1qmpB	136	439.95	59.65	1ntr_ 3chy_	130	339.86	5.53
1	1b00A 1qmpC	139	263.81	1.68	1ntr_ 4tmyA	126	450.05	3.34
1	1b00A 1qmpD	137	181.23	1.89	1ntr_ 4tmyB	127	399.26	3.75
1	1b00A 3chy_	154	141.50	0.85	1qmpA 1qmpB	221	3.77	0.03
1	1b00A 4tmyA	155	143.92	0.9	1qmpA 1qmpC	232	0.35	0.02
1	1b00A 4tmyB	155	75.41	0.73	1qmpA 1qmpD	230	0.02	0.03
1	1dbwA 1nat_	157	226.42	1.51	1qmpA 3chy_	160	69.78	1.07
1	1dbwA 1ntr_	130	426.13	5.53	1qmpA 4tmyA	162	98.21	0.78
1	1dbwA 1qmpA	152	159.74	2.93	1qmpA 4tmyB	164	50.48	0.62
1	1dbwA 1qmpB	150	63.63	1.52	1qmpB 1qmpC	221	1.60	0.02
1	1dbwA 1qmpC	150	180.52	2.38	1qmpB 1qmpD	220	1.61	0.03
1	1dbwA 1qmpD	152	111.28	1.78	1qmpB 3chy_	156	68.17	0.84
1	1dbwA 3chy_	164	84.22	1.19	1qmpB 4tmyA	157	51.32	0.58
1	1dbwA 4tmyA	161	73.71	1.1	1qmpB 4tmyB	156	66.11	0.64
1	1dbwA 4tmyB	163	47.87	1.11	1qmpC 1qmpD	226	3.65	0.02
1	1nat_ 1ntr_	127	302.39	3.59	1qmpC 3chy_	157	75.14	1.23
1	1nat_ 1qmpA	157	66.03	1.04	1qmpC 4tmyA	162	55.46	1.26
1	1nat_ 1qmpB	149	69.00	0.99	1qmpC 4tmyB	162	78.52	0.58
1	1nat_ 1qmpC	152	73.53	1.07	1qmpD 3chy_	158	59.47	1.11
1	1nat_ 1qmpD	151	99.14	1.33	1qmpD 4tmyA	157	59.23	0.71
1	1nat_ 3chy_	163	76.95	0.86	1qmpD 4tmyB	159	53.27	0.59
1	1nat_ 4tmyA	175	15.58	0.28	3chy_ 4tmyA	171	54.33	0.55
1	1nat_ 4tmyB	172	19.06	0.37	3chy_ 4tmyB	174	41.43	0.5
1					4tmyA 4tmyB	230	0.02	0.02
2	1bawA 1byoA	152	11.59	0.25	1byoB 2b3iA	135	7.21	0.27
2	1bawA 1byoB	155	6.11	0.18	1byoB 2pcy_	175	2.28	0.05
2	1bawA 1kdi_	140	33.84	0.55	1byoB 2plt_	174	3.90	0.06
2	1bawA 1nin_	153	9.45	0.21	1kdi_ 1nin_	129	52.53	1.13
2	1bawA 1pla_	124	28.04	0.62	1kdi_ 1pla_	126	33.59	0.89
2	1bawA 2b3iA	130	15.57	0.38	1kdi_ 2b3iA	122	40.83	0.84
2	1bawA 2pcy_	148	6.91	0.16	1kdi_ 2pcy_	145	15.19	0.3
2	1bawA 2plt_	161	5.22	0.13	1kdi_ 2plt_	150	24.56	0.32
2	1byoA 1byoB	192	2.61	0.02	1nin_ 1pla_	130	22.76	0.69
2	1byoA 1kdi_	148	17.89	0.35	1nin_ 2b3iA	129	25.55	0.5
2	1byoA 1nin_	140	30.14	0.85	1nin_ 2pcy_	139	23.31	0.49
2	1byoA 1pla_	150	7.55	0.16	1nin_ 2plt_	146	18.85	0.52
2	1byoA 2b3iA	132	10.26	0.39	1pla_ 2b3iA	122	12.65	0.32
2	1byoA 2pcy_	176	2.18	0.04	1pla_ 2pcy_	143	4.75	0.14
2	1byoA 2plt_	172	3.77	0.07	1pla_ 2plt_	144	7.10	0.17
2	1byoB 1kdi_	152	11.89	0.21	2b3iA 2pcy_	127	11.79	0.35
2	1byoB 1nin_	141	21.05	0.6	2b3iA 2plt_	140	7.37	0.17
2	1byoB 1pla_	148	6.94	0.16	2pcy_ 2plt_	172	3.67	0.06
3	1amk_ 1aw2A	411	1272.28	1.48	1btmA 1tmhA	432	1801.97	2.81
3	1amk_ 1b9bA	400	1044.23	2.04	1btmA 1treA	433	1512.26	2.59
3	1amk_ 1btmA	427	1287.48	2.38	1btmA 1tri_	419	1455.08	3.26
3	1amk_ 1htiA	407	265.16	1.4	1btmA 1ydvA	385	692.72	1.52
3	1amk_ 1tmhA	424	638.26	1.29	1btmA 3ypiA	406	1425.09	2.43
3	1amk_ 1treA	411	716.51	1.52	1btmA 8timA	408	940.59	2
3	1amk_ 1tri_	445	447.54	0.97	1htiA 1tmhA	416	588.98	1.07
3	1amk_ 1ydvA	384	462.44	1.05	1htiA 1treA	426	395.23	0.81
3	1amk_ 3ypiA	412	427.66	0.97	1htiA 1tri_	412	779.84	1.55
3	1amk_ 8timA	410	386.73	0.94	1htiA 1ydvA	382	405.04	1.09
3	1aw2A 1b9bA	411	961.04	3.28	1htiA 3ypiA	422	148.75	0.56

3	1aw2A 1btmA	434	750.67	3.1	1htiA 8timA	463	112.65	0.52
3	1aw2A 1htiA	425	363.03	1.78	1tmhA 1treA	513	119.27	0.23
3	1aw2A 1tmhA	474	185.72	0.51	1tmhA 1tri_	413	630.57	2.19
3	1aw2A 1treA	492	157.79	0.37	1tmhA 1ydvA	384	785.56	1.5
3	1aw2A 1tri_	408	1313.53	3.51	1tmhA 3ypiA	417	766.79	2.11
3	1aw2A 1ydvA	386	650.55	1.62	1tmhA 8timA	421	516.44	1.47
3	1aw2A 3ypiA	401	895.17	2.28	1treA 1tri_	401	1169.41	2.68
3	1aw2A 8timA	423	276.06	1.76	1treA 1ydvA	389	1419.90	2.21
3	1b9bA 1btmA	441	653.29	2.08	1treA 3ypiA	407	522.65	1.34
3	1b9bA 1htiA	394	809.23	2.27	1treA 8timA	425	310.95	1.15
3	1b9bA 1tmhA	418	548.56	1.34	1tri_ 1ydvA	371	1040.31	1.92
3	1b9bA 1treA	410	613.99	1.25	1tri_ 3ypiA	412	607.52	1.75
3	1b9bA 1tri_	391	1804.98	3.32	1tri_ 8timA	412	830.38	1.45
3	1b9bA 1ydvA	362	1608.97	6.1	1ydvA 3ypiA	374	355.82	0.92
3	1b9bA 3ypiA	396	700.45	1.88	1ydvA 8timA	388	399.47	0.99
3	1b9bA 8timA	392	634.48	1.66	3ypiA 8timA	418	267.14	0.65
3	1btmA 1htiA	403	1566.88	3.51				
4	1b71A 1bcfA	211	1800.08	453.08	1bcfA 1rcd_	222	528.84	1.99
4	1b71A 1dpsA	174	1800.43	266.54	1dpsA 1fha_	180	1800.24	9.45
4	1b71A 1fha_	216	1802.46	303.02	1dpsA 1ier_	184	1800.31	8.42
4	1b71A 1ier_	214	1801.32	480.43	1dpsA 1rcd_	184	1490.02	5.7
4	1b71A 1rcd_	211	1802.48	319	1fha_ 1ier_	299	69.34	0.25
4	1bcfA 1dpsA	187	510.17	3.81	1fha_ 1rcd_	295	36.40	0.19
4	1bcfA 1fha_	218	1017.59	2.69	1ier_ 1rcd_	297	24.03	0.15
4	1bcfA 1ier_	226	556.33	3.28				
5	1rn1A 1rn1B	191	1.23	0.03	1rn1B 1rn1C	197	0.21	0.01
5	1rn1A 1rn1C	190	1.01	0.03				
6	1qmpD 1tri_	131	1801.09	1674.98	1byoB 1rn1C	66	1800.09	686.03
6	1kdi_ 1qmpD	73	1800.15	904.75	1dbwA 1treA	145	1802.01	1703.2
6	1tmhA 4tmyB	112	1802.80	1521.23	1dbwA 1tri_	149	1800.73	1173.5
6	1dpsA 4tmyB	89	1800.39	913.24				

Table 2: Column one contains the number of the families according to table 1. The sixth class contains the hardest solved Skolnick set intstances. Column two(six) contains the names of the couples, column three(seven) is the score, column four(height) gives the time in seconds taken by LR algoritm, and column five(nine) presents the corresponding time taken by a_purva.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399