Taxonomy and survey of scientific workflow scheduling in infrastructure-as-a-service cloud computing systems

Fouakeu-Tatieze Stéphane, Kamla Vivient Corneille, Sonia Yassa, Kamgang Jean-Claude, Nkenlifack Marcellin Julius Antonio

Journal of Advanced Computer Science & Technology, 12 (1) (2024) 19-32

Journal of Advanced Computer Science & Technology

Website: www.sciencepubco.com/index.php/JACST

Research paper

Taxonomy and survey of scientific workflow scheduling in infrastructure-as-a-service cloud computing systems

Fouakeu-Tatieze Stéphane [1] *, kamla Vivient Corneille [2], Sonia Yassa [3], Kamgang Jean-Claude [4], Nkenlifack Marcellin Julius Antonio [5]

[1] Department of Electrical Engineering and Industrial Automation ENSAI University of Ngaoundere Cameroon
[2] Department of Mathematics and Computer Science ENSAI University of Ngaoundere Cameroon
[3] ETIS Laboratory UMR8051 CNRS CY Cergy Paris University France
[4] Department of Mathematics and Computer Science ENSAI University of Ngaoundere Cameroon
[5] Department of Mathematics and Computer Science Faculty of Science University of Dschang
**Corresponding author E-mail: fouakeustephane@gmail.com*

Abstract

Scientific workflows are groups of scientific application tasks organized in oriented graphs. These scientific workflows are characterized by a large number of tasks requiring sufficient resources for their execution. Tasks with all available input data can be computed simultaneously. Cloud computing is an appropriate environment for the implementation of scientific workflows. Although the cloud computing environment has unlimited resources and can run some scientific workflow tasks simultaneously, scheduling scientific workflow tasks using pay-as-you-go cloud computing resources is an NP-complete problem. This difficulty is due to constraints on the part of the cloud resource provider and the part of the user (customer). The algorithm tries to find efficient schedules that take into account several requirements of client service (QoS) such as deadlines, budgets, and resource providers' profits, i.e., the minimization of energy consumption and many others. There have been several papers recommending effective solutions to workflow schedule problems. This paper reviews existing and more recent papers on the plan of scientific workflows in pay-as-you-go IaaS cloud computing environments, focusing on future directions for algorithms that can improve the optimal solution.

Keywords: *Scientific Workflows; Scheduling Algorithms; IaaS Cloud; Taxonomy; Survey.*

1. Introduction

Scientific workflows are commonly used to model a scientific application [1]. They describe a set of processing that allows data to be analyzed in a structured and distributed way and have been used to achieve significant scientific advances in various fields. The emergence of Cloud Computing has brought many benefits to the deployment of large-scale scientific workflows [2]. In particular, Cloud Computing as a service resource provides an easily accessible, flexible, and scalable infrastructure for deploying scientific workflows [3]. Infrastructure as a service (IaaS) is a cloud service model that offers essential computing, storage, and networking resources on demand, on a pay-as-you-go basis. IaaS, along with Software as a Service (SaaS) and Platform as a Service (PaaS), is one of the three major categories of cloud computing services. IaaS cloud providers offer the ability to run workflows on their infrastructure by leasing compute resources, storage resources, and more [4]. These resources are billed based on usage [5]. Thus, the resource usage of a workflow can be adjusted according to the desired execution time or the budgeted resources to be used [6]. Scheduling algorithms are essential to take advantage of the benefits of cloud computing to efficiently automate the execution of scientific workflows in distributed environments [7]. These algorithms are an essential component of workflow management systems that are responsible for orchestrating the execution of tasks across a set of computing resources while preserving data dependencies. The decisions made by these scheduling algorithms are usually guided by a set of user-defined Quality of Service (QoS) requirements. A scientific workflow in a distributed system is often scheduled by allocating tasks to resources and coordinating their execution in a way that maintains dependencies. [8].
The mapping is also done in such a way that different user requirements are met. The scheduling objectives are determined by these criteria, which are typically expressed in terms of performance indicators like execution time [9], [10], service utilization cost [11], and non-functional requirements such as security and energy consumption [12]. Two smaller issues need to be taken into account when planning how a workflow will be carried out in a cloud computing environment [13], [14]. The first is known as resource provisioning, which consists of selecting and procuring the computing resources that will be used to execute each task in the scientific workflow. Thus a fundamental question arises, namely how to find the right configuration of resources needed to run a scientific workflow? This means having methods capable of determining the number of computing cores (virtual machines) to rent. The second issue pertains to the job allo-

cation or scheduling phase, wherein every task is assigned to the most appropriate resource. Some authors in the literature creating algorithms for cloud resources frequently refer to the combination of these two sub-problems as "scheduling" [15], [16].
In this paper, we look at various recent algorithms for IaaS cloud computing's scientific workflow scheduling. This study concentrated on workflow scheduling approaches, how workflows enter the scheduler, how to provision cloud resources into the scheduler, QoS measurements, and cloud resource usage. Readers will have a better understanding of the issue of scheduling scientific workflows in cloud computing and new directions for future study as a result of the examination of the various classifications established in this article.
The rest of the document is structured as follows: Section 2 introduces scientific workflows as a whole, and Section 3 shows how to plan scientific workflows in a cloud computing environment. We have the survey in Section 4, which presents some recent scientific workflow scheduling algorithms studied in the literature. Section 5 first presents a table that classifies the algorithms studied according to certain targeted characteristics and then presents the discussion. Before the conclusion of Section 7, we will have, in Section 6, some open issues. Figure 1 shows the organization of the paper.

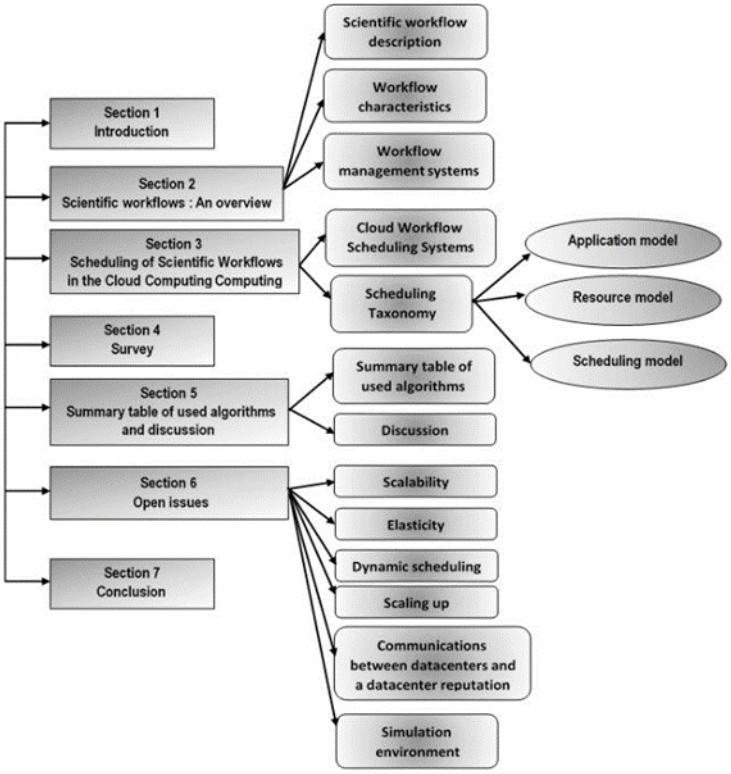

Fig. 1: Flow of the Article.

2. Scientific workflows: an overview

The notion of workflow first appeared in the electronic imaging and computer-aided management industry [17], in order to automate work processes within organizations. In addition, today's processing chains (workflows) benefit from a runtime environment that is able to guarantee compliance with the service level agreement (SLA) between the providers of the resources in that environment and its customers while guaranteeing QoS and provider gains. There are two main categories of uses that use the notion of workflow: experimental protocols in fields such as biology, astronomy, physics, neuroscience, chemistry, etc. and the processing chains used in commercial, financial, and pharmaceutical domains (called business processes). They give rise to several diverse but related, lines of research. In this thesis, we focus on scientific workflows. According to the WfMC (Workflow Management Coalition) [18], we can retain the following definition of workflow independently of the specific domains:" Workflows are automated business processes that can be completed entirely or in part. They involve the transfer of papers, data, or tasks from one participant to another for processing in accordance with protocol guidelines". However, several definitions have been proposed according to the categories of use, so with regard to the scientific workflow we will retain the following definition: A scientific workflow is the description of a process to achieve a scientific goal, generally consisting of a set of tasks and data dependencies between them. In scientific workflows, tasks are typically data analysis steps or computing procedures for scientific simulations. Acquisition, integration, reduction, visualization, and publication (e.g., in a shared database) of scientific data are common components or processes in scientific workflows. A scientific workflow's actions are arranged (during design time) and coordinated (during run time) according to the data flow and any additional dependencies that the workflow designer has indicated.

2.1. Scientific workflow description

In order to automate processing and save time, scientists very often need to gather software tasks and interconnect them to form an application. At the entrance of the application, data begins to be processed by the tasks which have no dependency on other tasks, then these transmit the results to the following ones. Generally, the structure of such an application is represented by a directed graph without cycle: DAG (Directed Acyclic Graph). In this graph, each node (vertex) is a task and the arcs are the dependency constraints. This addiction also models communication between two tasks. In the literature, the notion of workflow is generally defined as an abstract structure of an application described by a DAG. Other models for describing workflows have existed for a few decades. Regardless of the workflow

model, research distinguishes the execution of an application equipped with a workflow structure that performs processing on a set of input data and the parallel execution of several tasks of the same application [19].

2.2. Workflow characteristics

Data-intensive parallel applications (including scientific workflows) are commonly used in the majority of disciplines often leveraging rich and varied data resources as well as parallel and distributed computing platforms. Workflows provide a systematic way to describe the methods needed to represent a parallel application. They act as a liaison between IT infrastructure experts and subject matter experts in the field. Scientific Workflow Management Systems (SWfMS) handle a range of dispersed resources and carry out intricate analyses. Workflows are becoming more and more important as a result of the sharp rise in data volumes and diversity across all fields. They enable researchers to develop processing and analysis strategies to draw conclusions from a variety of data sources and take advantage of a vast array of computing and data platforms. To carry out a study, scientific workflows are very often modeled in the form of DAGs as mentioned above. A scientific workflow, as shown in Figure 2, is a set of nodes and arcs. The nodes represent the tasks of the workflow and the arcs between the different tasks represent either a data dependency, mainly a transfer of data, or a flow dependency between two tasks. Each of the tasks composing the scientific workflow has an estimated duration, requires an input dataset to start its execution, and produces an output dataset at the end. When output data produced by one task is consumed as input by another, this creates a data dependency between the two tasks. The dependency between the two is represented by a directed arc in the DAG. Input data that is not generated by any of the workflow tasks is called scientific workflow input data. Conversely, output data that is not consumed by a task is called scientific workflow output data.

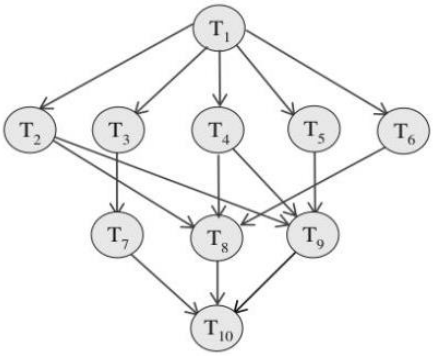

Fig. 2: A DAG Representing A Sample 10-Task Workflow [20].

2.3. Workflow management systems

Scientific workflows are used to model complex applications in DAG (Directed Acyclic Graph) format with nodes and arcs that easily express the entire data process with its dependencies [21]. When multiple data (input and output) are consumed and produced during science experiments, this makes workflows data-intensive. As the complexity of scientific applications increases, the need for using Scientific Workflow Management Systems also increases to automate and orchestrate end-to-end processing (execution). In order to process data at scale, they must be run in a distributed environment such as the cloud. The workflow management system is an efficient framework for managing massive data sets in a computing environment (grid or cloud). Nowadays, there are several workflow management systems for grid and cloud computing, of which the most popular are: KEPLER [22], TAVERNA [23], TRIANA [24], ASKALON [25], PEGASUS [26]. Workflow management is mainly the coordination of the tasks that constitute the organizational whole of the workflow. Thus, the workflow management system is mainly the set of tools essential for better organization (defining, mapping, and executing) tasks while relying on the different data inputs and outputs (as shown in Figure 2).

KEPLER [22] is a cross-project collaboration to develop a scientific workflow management system in which scientific workflows can be designed and executed. KEPLER is an open-source Java framework developed and maintained by the Kepler community and is a derivative of Ptolemy [27]. It is designed to help the scientific community analyze and model a wide range of scientific applications. This framework supports the visual representation of processes. Using a visual representation simplifies the creative effort.

TAVERNA [23] is an open-source, Java-based workflow management system created by the myGrid team that designs and executes scientific workflows. The main objective of Taverna is to extend support to the field of life sciences, chemistry, and medicine, in order to run scientific workflows and support experimentation on silicon, where experiments are performed by computer simulation and closely mirror the real world. It supports web services, Java application programming interfaces (APIs), R scripts, and tabular data files (CSV).

TRIANA [24] is a scientific workflow management system based on the Java language and uses a graphical interface with data analysis tools. TRIANA has various built-in tools for image manipulation, signal analysis, and more, allowing researchers to integrate their own tools. To run the workflows, TRIANA has a workflow engine called Triana Controlling Service (TCS).

The ASKALON project [25] is developed by the University of Innsbruck (USA). It provides an ideal environment based on new services, tools, and methodologies for running parallel applications in cloud and grid environments. Its goal is to simplify the development and optimization of parallel applications. Workflows are described using AGWL (Abstract Grid Workflow Language). It allows researchers design applications using modeling instead of programming and optimize parallel applications based on constraints (dependencies).

PEGASUS [26] is an open-source workflow management system developed at the University of Southern California, which integrates a number of technologies to run parallel applications in heterogeneous environments such as grids, clusters, and clouds. It has been used in a number of scientific fields such as bioinformatics, gravitational wave physics, ocean science, etc.

One of the most important elements of workflow management systems is the scheduling module, which we will present in the next section.

3. Scheduling of scientific workflows in the cloud computing

3.1. Cloud workflow scheduling system

Fig. 3 shows the architecture of the cloud workflow scheduling system. We can see that the system consists of four modules: an estimation module, a scheduling module, a reservation module, and an execution module. To schedule a scientific workflow onto an IaaS cloud, seven steps should be taken [28]:

1) At first, the estimation module acquires the workflow specification and the QoS requirement from the user, and the virtual machines (VM) specification from the cloud service provider.
2) Based on the acquired information, the estimation module will estimate the execution time of each task on every kinds of VM.
3) Then, taking the execution time matrix, the scheduling module will make a complete execution plan about how many VMs should be leased, when to lease and release them, and which task should be assigned to which VM. The scheduling module consists of two components: optimization algorithm and simulation. The optimization algorithm generates schedules and gives them to the simulation component to judge whether the schedules can satisfy the constraints and how good the schedules are. Then utilizing the simulation results, the optimization algorithm keeps finding better schedules until the stop criterion is met.
4) After getting a near-optimal schedule, the scheduling module tells the reservation module the amount and the types of the required VMs and gives the execution plan to the execution module.
5) The reservation module then leases VMs from the service provider and prepares them ready.
6) When the execution module is informed that the runtime environment is prepared, it starts sending commands to the cloud to execute the workflow, telling the cloud which VM should be started or power-off.
7) During the execution, it collects the information about the real runtime of tasks and returns such information to the estimation module as feedback. If there is another similar workflow that is going to be executed, the feedback information can help in improving the precision of execution time estimation. After running the whole workflow, the execution module is also responsible for returning the final result to the user. Among these four modules, the former two, estimation module and scheduling module, are the core of the whole scheduling system, because they together decide whether an effective and economical schedule can be generated. Workflow scheduling is a process that ”maps” and manages the execution of interdependent tasks onto distributed resources. It allocates appropriate resources for workflow tasks so that execution can be completed while satisfying user-imposed goals and constraints. Proper scheduling can have a significant impact on system performance. In general, the task mapping problem on distributed services belongs to a class of problems known as NP-complete [29], [30].

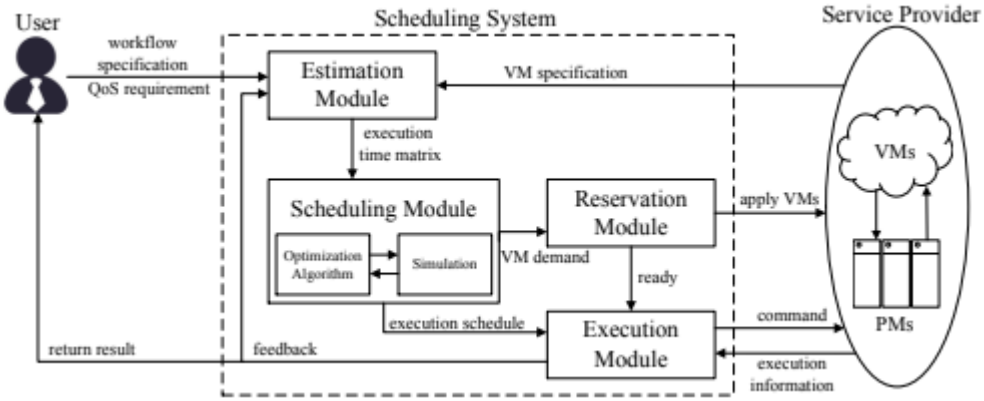

Fig. 3: Architecture of the Cloud Workflow Scheduling System [28].

3.2. Scheduling taxonomy

The algorithms developed in this survey focus only on IaaS clouds. This part aims to give a clear explanation for each model presented and used in the classification table of Section 5. These are the application model, resource model, and scheduling model.

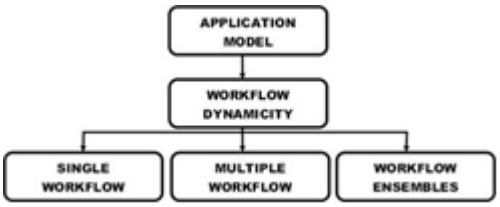

Fig. 4: Application Model Taxonomy.

3.2.1. Application model

Users submit requests to service providers as a workflow. However, applications have the ability to be planned and executed on the IaaS cloud based on their dynamics. The purpose of dynamicity for workflow scheduling algorithms is to manage the execution of a single workflow, multiple workflows, or sets of workflows.

a) Single workflow

In the literature, several algorithms, in the case of articles [5] [31 - 48] focus on the independent and sequential scheduling of scientific workflows that arrive at different times.

The class’s algorithms are made to optimize a single workflow’s schedule. This is still the most often used paradigm in cloud computing; it is the one from the past that is utilized in grids and clusters. It is assumed that the scheduler controls the independent and sequential

execution of workflows. The scheduling algorithm can therefore concentrate on maximizing the quality of service needs for a single user and a single DAG [49].

b) Multiple workflows

Although the workflows being scheduled in this category are comparable to those in the workflow ensembles category, they are not necessarily related to one another and may differ in terms of structure, size, input data, application, etc. More crucially, scheduling is seen as a dynamic process where workflows with different configurations are continuously arriving for execution and the workload is constantly changing since the quantity and kind of workflows are unknown in advance. One more distinction is that every workflow instance has distinct QoS requirements of its own [49].

The algorithms in [50 - 52] each schedule, at the same time, several scientific workflows of different sizes, which arrive in the platform for execution. In this scheduling, the objective is to meet the quality of service requirements demanded by each submitted workflow. For example, [51] in particular proposes a multi-objective optimization model for several workflows, taking into account the fairness of each workflow through the clustering method, and presents cluster-based resource allocation approaches to improve the optimization.

c) Workflow ensembles

Many scientific applications consist of multiple instances of the procedure. These connected processes are referred to as ensembles and are grouped because when they are executed collectively, they yield the intended result [53]. While the quantity and input data of the workflows in an ensemble vary, they often share a similar structure. The goal of scheduling algorithms is to use the resources at hand to carry out each workflow on the ensemble. The fact that the QoS requirements are intended for numerous workflows rather than just one must be understood by policies. For instance, an ensemble with 100 workflows and a 1-hour deadline must finish all of the workflows before the deadline. Because of this, algorithms typically include the quantity of work (i.e., the number of workflows that have been finished) in their scheduling objectives. Another characteristic of workflow ensembles is that the number of instances is usually known in advance, and therefore the planning strategy can use it when executing tasks [49].

In the literature, few algorithms are interested in scheduling Workflow ensembles. The EMO [54] algorithm, for example, is based on a meta-heuristic like the genetic algorithm to schedule Workflow ensembles.

3.2.2. Resource model

For the resource model, the algorithms adopt a resource provisioning strategy which can be static or dynamic. In the static resource provisioning strategy, all decisions regarding the configuration of the virtual machine pool are made prior to the execution of the workflow. For the dynamic resource provisioning strategy, on the other hand, all initial decisions at runtime are made by selecting which VMs to keep active, which to lease and which to release as the workflow is executed [49]. Figure 4 shows the different types of resource provisioning strategies.

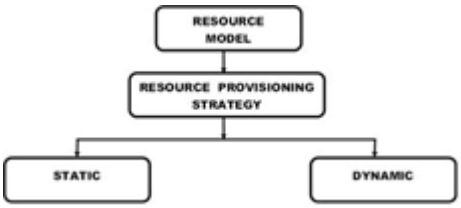

Fig. 5: Resource Model Taxonomy.

a) Static

In static resource provisioning, the resources are rented and kept active during the workflow's execution after the VM pool has been established. The resources are given back to the supplier when the application expires. Estimating the capacity of the resources required to satisfy the scheduling goals is the main focus of these methods. The benefit is that the algorithm may concentrate entirely on assigning jobs to virtual machines (VMs) once the resource provisioning decision has been made. The effects of VM provisioning and deprovisioning delays are greatly dampened and become much more manageable. Nevertheless, this model disregards the cloud pricing paradigm and does not capitalize on resource elasticity. Because VMs will be billed even during periods when they are idle, this might lead to schedules that are not cost-effective and do not meet QoS requirements [49].

Several algorithms [54 - 56], [31], [34], [35], [37 - 40], [43], [44], [46,] work with static resource provisioning. Before starting the actual scheduling, the number of resources needed is already known in advance. The reservation of resources can therefore be done with the cloud infrastructure provider.

b) Dynamic

This tactic works well with algorithms that use either dynamic or static resource provisioning. With this approach, algorithms can be updated as the workflow is carried out, both in terms of the quantity and kind of virtual machines (VMs) utilized to schedule activities. Certain algorithms are able to make adaptive choices in response to task restrictions and cost awareness. For example, idle virtual machines (VMs) can be shut down to save money, and a fresh VM can be deployed so that the job being planned can finish before its deadline. An additional method to accomplish this would be to estimate the capacity of resources required by activities on a regular basis to fulfill the limitations of the application and modify the virtual machine pool accordingly. Some methods rely their scaling decisions on performance measures like task throughput and overall virtual machine utilization. For instance, if the budget permits it and the utilization increases over a predetermined level or if the quantity of tasks completed per second falls below a predetermined threshold, more virtual machines (VMs) may be provisioned. Finally, while creating the static schedule, static algorithms that make advantage of elastic virtual machine pools do so by ascertaining the VMs' lease terms. The projected start time of the first task issued to a virtual machine (VM) and the estimated finish time of the last work assigned to it set limits on these leasing periods [49].

Some algorithms in [33], [32], [5], [36], [41], [42], [45], [50], [52] use the dynamic provisioning strategy. At the beginning of application scheduling, the number of resources to be used remains unknown. It is only during the scheduling process that the need for the number of resources arises until the end.

3.2.3. Scheduling model

In the scheduling system of scientific workflows, many studies have been done in the IaaS environment of cloud computing. For these studies, the different authors define a scheduling model. In this model, two strategies emerge: the optimization strategy and the scheduling objectives, which are the different metrics to be considered in the optimization. Figure 5 shows the general scheduling model used in the scheduling process of workflows in the cloud.

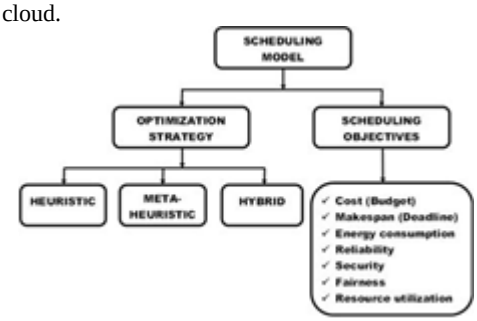

Fig. 6: Scheduling Model Taxonomy.

a) Optimization strategy

- *Heuristic*

Numerous strategies for optimizing scientific workflows based on heuristics have been proposed such as BAGS [31], PDPPS [32], BDAS [33], FDHEFT [34], GRP-HEFT [36], NBWS [37], CTTSW [38], CTTWSDP [41], SRPSM [50], BDDC+BDC [42], MOWOS [56], MHPSLP [43] [51], SMWDSA [52], REWS [46], EViMA [45]. Heuristics are generally a collection of guidelines designed to solve certain problems [57]. These guidelines are particular to the issue at hand and are made to ensure that a rough solution is found in a reasonable amount of time. In the scheduling scenario under discussion, a heuristic technique makes use of the workflow application's features and the cloud's attributes to find a timetable that satisfies the user's quality of service (QoS) needs. Heuristic-based scheduling algorithms' primary benefit is their performance efficiency; they typically discover workable solutions in a reasonable amount of time. In comparison to meta-heuristic-based approaches, they are also more predictable and simpler to use [49].

- *Meta-heuristic*

Examples of meta-heuristics applied to scheduling scientific workflows in the cloud are the genetic algorithm, GA [54, 39], and particle swarm optimization PSO [44]. While heuristics are specifically tailored to address a given problem, meta-heuristics are all-purpose algorithms that address optimization issues [57]. These are higher-level tactics that use heuristics particular to a certain problem to identify a close to ideal solution. Meta-heuristic techniques are often more computationally costly and take longer to execute than heuristic-based algorithms, but they also have a tendency to find more desirable schedules as they use a guided search to investigate many possibilities. Defining operations to prevent exploring invalid solutions (e.g., data dependency violations) to facilitate convergence, modeling an essentially unbound number of resources, and pruning the search space using heuristics based on the cloud resource model are some of the challenges involved in using meta-heuristics to solve the workflow scheduling problem in clouds [49].

- *Hybrid*

To optimize the scheduling of a collection of workflow tasks, hybrid algorithms may employ meta-heuristic techniques. Using heuristics, one further approach is to integrate optimal solutions for smaller and/or simplified versions of the problem. Because algorithms analyze a narrower problem area than heuristic-based methods, they may be able to make better optimization decisions while requiring less computing time [49]. Some works use the hybrid strategy for scheduling workflows, such as [35] which combines AG and ABC, [40] which combines HEFT and ACO, [55] with PSO and GWO.

b) Scheduling objectives

Much of the work focuses on optimizing the metrics presented below:

- *Cost*

Minimization of costs: Cloud platform algorithms must take infrastructure leasing into account. Should they neglect to do so, using cloud storage, renting virtual machines, and moving data might get quite costly. Algorithms incorporate this goal by either attempting to lower its value or by placing a limit on the total amount of money allocated to resources, primarily budget [49]. The subsequent equation might be used to convey it [58]:

$$Cost = Cost_{ex} - Cost_{tr} \quad (1)$$

Where $Cost_{ex}$ is the execution cost and $Cost_{tr}$ is the transfer cost.

- *Budget*

It is the cost that the consumers pay for the usage of the cloud resources [59].

- *Makespan*

Makespan minimization: the makespan of a workflow is the time at which the last workflow task finishes its execution [60]. In order to calculate it, use the following equation [61]:

$$Makespan = FTime - STime \quad (2)$$

Where FTime denotes the Finish time of the last task and STime is the starting time of the first task.

- *Deadline*

It is the time limit for the execution of the workflow [62] and its supporting is an important QoS requirements.

- *Energy consumption*

Minimizing energy consumption: To minimize their negative effects on the environment, people, businesses, and governments all over the world are becoming more concerned about lowering their carbon footprints [49].

- *Reliability*

Reliability awareness: it is the probability that the task can be completed successfully. To achieve this, scheduling mechanisms like active replications and backup/restart plans that account for resource and temporal redundancy may be used [63].

- *Security*

Security awareness: attackers may misuse some cloud features and components to launch cloud-specific attacks [64]. A secure scheduler produces safe scheduling to mitigate the effects of security attacks.

- *Fairness*

Li et al. [51] define a new metric for fairness. Fair resource allocation across various processes is a genuine challenge for multi-workflow scheduling, albeit it goes beyond costs and makespan. The slowness that each workflow would encounter when sharing resources with other workflows is the basis for defining fairness.

- *Resource utilization*

VM Utilization Maximization: Algorithms attempt to eliminate idle time slots in their schedules since they are considered a waste of money in leased virtual machines (VMs) because they were paid for but not used. But these wasted time slots frequently result from a workflow's execution, mostly because of job dependencies and performance standards [49].

4. Survey

In 2016, Zhu et al. [54] proposed an evolutionary multi-objective optimization algorithm to solve the workflow scheduling problem in the cloud IaaS infrastructure by simultaneously minimizing the makespan and the execution cost. This algorithm, based on the genetic algorithm, is a meta-heuristic consisting of a system for coding solutions, initializing the population, evaluating the fitness function, and applying new genetic operators. Simulation results show that this algorithm has the best solutions compared to other scheduling algorithms for QoS optimization in the literature.

In 2017, Rodriguez and Buyya [31] proposed a budget-based algorithm called BAGS in which different resource sourcing and scheduling strategies are used for different topological structures. This algorithm consists of four main steps. The first is an offline strategy that partitions the DAG into BoTs before it is executed. The second is an online budget distribution phase that is repeated throughout the execution of the workflow. It allocates part of the remaining budget to tasks that have not yet been scheduled. The third step is responsible for creating a resource supply plan for the BoTs as their tasks become available for execution. Finally, the ready tasks are scheduled and executed according to their corresponding supply plans. The simulation results demonstrate the responsiveness of the proposed algorithm to environmental uncertainties and its ability to generate high-quality schedules that meet the budget constraint while achieving faster execution times compared to state-of-the-art algorithms.

Singh et al. [32] in 2018, proposed a Partition Problem-based Dynamic Provisioning and Scheduling (PDPPS) algorithm to reduce the execution cost of scheduling a time-bound workflow using dynamic resource provisioning in a single cloud environment. This algorithm has two phases. A first phase is an approach that uses a delay determination method for each bot and the k-means clustering technique to determine the speed of the virtual machines that would be used in the scheduling. The second phase is a dynamic VM provisioning approach for task-to-VM mapping, using the partition problem, which is a variant of the subset-sum problem. They incorporate a model used by many cloud service providers today, based on centralized storage for data transfer, that can help with data recovery in the event of a virtual machine failure in the middle of a task execution. They establish that PDPPS works better than existing approaches such as IC-PCP and DPDS via simulations.

In 2019, Arabnejad et al., [33] presented a new heuristic scheduling algorithm - Budget and Deadline Aware Scheduling (BDAS) to schedule both budget and deadline-constrained workflows in Infrastructure as a Service (IaaS) clouds. In order to find the most feasible scheduling and the best instance type to furnish, the BDAS algorithm employs a novel time-cost trade-off factor. BDAS is divided into five main phases: Scientific Workflow division, allocation of funds, deadline distribution, job selection, and virtual machine instance selection. BDAS was assessed and contrasted with three previously published algorithms (BDHEFT, RCT, and RTC) based on a variety of measures. In terms of success rate, the BDAS algorithm shows over 84% success in all datasets, while the worst performance occurs in the BDHEFT algorithm which fails in about 60% of different test cases in EPIGENOMICS and LIGO.

Zhou et al. [34] in 2019, focus on the scientific workflow scheduling problem of simultaneously minimizing cost and time under task precedence constraints in the workflow. They design a heterogeneous fuzzy dominance sort (FDHEFT) algorithm to solve the cloud workflow scheduling problem. Just like MOHEFT, FDHEFT is an improved form of HEFT that is separated into two primary stages: the tasks of the prioritizing phase and the instance selection phase. The scheduling priorities of every activity in the workflow are assigned during the task prioritizing phase. The optimal instance for every task in the schedule list is then found during the instance selection phase. The extensive experiments are based on actual Amazon EC2 pricing and resource parameters, and the results demonstrated that the proposed FDHEFT can explore better trade-offs between cost and workflow scheduling with significantly lower execution times compared to a number of similar approaches.

In 2019, Gao et al. [35] developed a Pareto-based multi-objective workflow scheduling algorithm, called HGAABC, to simultaneously optimize the makespan and execution cost of the workflow. They extend HGAABC to cope with commercial IaaS cloud computing systems providing a limited number of instances and a flexible mix of instance types. In terms of the pay-per-use price model, they translate each job into a matching VM instance series type by integrating the operation and exploration capabilities in ABC and GA, respectively. To produce a workable schedule given a mapping of tasks to virtual machines (VMs) and a mapping of VMs to their types, a decoding heuristic is provided. These mappings are evolved by a hybrid algorithm that combines the principles of GA and ABC. In the GA algorithm, the solutions are called chromosomes which are recursively improved by the selection, crossover and mutation operators. In the second half of the iteration cycle, the resultant chromosomes are sent to the ABC algorithm. Worker bees, spectator bees, and scout bees are the three kinds of bees that gradually enrich the solutions, which are referred to as food sources in the ABC algorithm.

In 2019, the authors of the paper [5] propose an onliNe multi-workflOw Scheduling Framework, named NOSF, for IaaS clouds. By dynamically allocating VM resources for randomly arrived workflows based on the online status of scheduled tasks, NOSF not only captures the stochasticity of task execution time on VMs but also mitigates the propagated impacts of the fluctuated task execution time on each VM. The scheduling process of workflows in NOSF consists of three phases: workflow preprocessing, VM allocation, and feedback process.

They conduct extensive simulations on realistic workflow datasets, and validate the superiority of NOSF, by comparing it to two existing algorithms, in terms of VM rental cost and deadline violation probability reduction (average 50.5 and 55.7 percent, respectively), as well as improving resource utilization efficiency (average 32.6 percent).

In 2020, Faragardi et al. [36] suggested the Greedy Resource Provisioning and Modified HEFT (GRP-HEFT) scheduling algorithm and method for resource provisioning for the effective execution of workflows, to reduce the lifetime of a certain scientific workflow within the hourly cost model's budgetary constraints. GRP-HEFT contains two parts: a resource provisioning algorithm that specifies how many and what types of instances are extracted from the infinite pool of resources offered by cloud providers and a scheduling algorithm that determines the assignment of tasks to the obtained instances and the order of execution of the tasks within each instance. They compare their algorithm with three different baseline algorithms that are considered state-of-the-art. According to the results of large-scale workflows with 1000 tasks, in all experiments, they strictly outperform these algorithms.

In 2020, Kalyan et al. [37] presented an algorithm called NBWS (Normalization-based Budget Constraint Workflow Scheduling) that generates a workflow schedule that minimizes execution time while respecting the budget constraint. To minimize execution time, the NBWS algorithm maps workflow tasks to resources that have the earliest finish time within the allocated budget. This budget-constrained algorithm consists of finding a viable workflow schedule subject to the user-defined budget. The proposed NBWS consists of two distinct stages, namely the task selection stage and the elastic resource selection stage. For the simulation, the CloudSim tool is used and the obtained results are compared with basic algorithms such as BDHEFT, BHEFT, and IC-Loss on various real-world scientific workflows.

Mboula et al. [38] in 2020, suggested a novel workflow scheduling technique with the goal of minimizing processing expenses and execution times: Effective Workflow Scheduling with a Cost-Time Trade-off (CTTWS). Implicit Requested Instance Types Range (IRITR) evaluation is a novel idea used by the CTTWS scheduling algorithm to identify a range of virtual machine instance types that best suit the workflow execution and prevent overbids and underbids, which can result in budget and deadline violations, respectively. As a result, no task uses a slower instance than those of the IRITR, and the root tasks are carried out on comparatively fast instances that speed up execution and may be reused. Based on their cost ratio, time ratio, and success rate, they contrast BDAS and CTTWS. This study shows that CTTWS is more effective than BDAS and outperforms BDAS by up to 38.4% in scheduling success.

In 2021, the authors [40] suggested using the ant colony algorithm (ACO) in conjunction with the heterogeneous earliest finish time (HEFT) to reduce both the makespan and the cost. The main goal of this work is to improve the process of allocating workflow jobs to virtual machines by using a metaheuristic algorithm. Planning a workflow often involves two stages. Setting a priority for reliant chores came first. The second is utilized to plan out tasks that are prepared. The outcomes show that HEFT-ACO is a viable strategy to deal with QoS problems related to cloud workflow resource allocation.

In 2021, the authors of [41] developed the CTTWSDP (Cost-Time Trade-off efficient Workflow Scheduling with Dynamic provisioning) algorithm. The latter uses a cost-time trade-off function on heterogeneous instances and dynamic virtual machine provisioning with a restricted number of rented virtual machines to identify the most practical scheduling. CTTWSDP also presented an improved evaluation of the Implicit Requested Instance Type Range (IRITR), which is a scheduling concept developed in their prior work [38]. The goal of the IRITR assessment is to identify a range of virtual machine instance types that are most appropriate for executing the workflow, preventing underbids and overbids that could violate deadlines and the budget, respectively. The simulations' outcomes demonstrate the proposal's efficacy. CTTWSDP achieves a higher success rate of 17.09-76.06% compared to four leading algorithms in the literature.

In 2021, Rajasekar et al. [50] suggested a resource provisioning and scheduling technique designed specifically for WaaS platforms. The method can be made more dynamic and scalable to enhance the platform and workload. It aims to lower the overall execution cost of infrastructure resources by meeting every workflow deadline constraint and allows containers to address inefficient resource use. According to what we know, this method specifically addresses virtual machine sharing in the context of cloud-as-a-service (WaaS) by implementing the use of containers in scheduling and resource provisioning heuristics. In comparison to other current algorithms, their testing results demonstrate its cost-effectiveness, responsiveness to environmental uncertainties, and ability to meet deadlines.

Taghinezhad-Niar et al. [42] suggested in 2021, two algorithms to account for time and money restrictions when developing scientific workflows on cloud IaaS platforms: Budget Deadline Delicate Cloud (BDDC) and Budget Deadline Cloud (BDC). BDDC is a heuristic algorithm that distributes budgets and deadlines for each level taking into account the overall task execution time to acquire economic costs for each level. Meanwhile, BDC distributes timelines to each level, and the budget is considered the remaining cost in each programming cycle. BDDC and BDC are suitable for low-budget environments. In general, BDDC and BDC run other algorithms providing higher success and higher quality of service and utilization.

In 2021 [55], the authors proposed an algorithm, named PSO–GWO, which is a combination of Particle Swarm Optimization and Grey Wolf Optimization. The basic idea of the PSO–GWO algorithm is to run the PSO algorithm to the first half of iterations and the best solution generated by the PSO (gbest) is initialized by the alpha wolf and for the later half run GWO algorithm. The best solution generated by the GWO is stored in alpha wolf and considered to be the best mapping of tasks and VMs. The PSO–GWO algorithm is tested on the scientific workflow like montage, cybershake, inspiral and sipht. The suggested approach seeks to minimize the overall cost of execution. The experiment's findings demonstrate that, when compared to the normal Particle Swarm Optimization and Grey Wolf Optimization algorithms, the PSO–GWO algorithm reduces both the average total execution time and cost.

In order to simultaneously decrease execution cost and execution makespan, Konjaang et al. [56] expanded their prior work "Cost Optimised Heuristic Method (COHA)" and developed a revolutionary workflow scheduling algorithm called Multi-Objective Workflow Optimization Strategy (MOWOS) in 2021. To shorten their scheduling time, MOWOS uses a task-splitting method to divide huge activities into smaller ones. Additionally, MOWOS introduces two new task allocation techniques dubbed MaxVM selection and MinVM selection. The goal of MOWOS's design is to make it possible for all tasks to complete their assignments on time and within budget. When compared to the state-of-the-art work, the suggested algorithm performs much better in large and extra-large workflow jobs than in small and medium workflow tasks.

The authors [43] in 2022 proposed a hybrid algorithm called MHPSLP which is an improvement of the article [65] with two different mathematical methods, they were put to use for scientific workflow scheduling and resource provisioning. They introduce a linear mathematical model to fairly distribute the delays on the tasks according to their execution time in each workflow level in the static phase, then they replace their provisioning method with a linear mixed-numbers programming method to select optimal combinations of resources to reduce workflow costs. The advantage of this method compared to the compared scheduling algorithms is the reduction of the cost of the task executed within a time constraint.

The authors of the paper [51] in 2022 focus on scheduling computing resources under multi-objective optimization for multiple workflows. In particular, these authors propose a multi-objective optimization model for multiple workflows considering the fairness of each workflow through the DFS-CST-based clustering method and present resource allocation approaches based on clusters to improve optimization. For this, they define the fairness of the multi-workflow scheduling considering the fairness of individual execution time and cost, then they design a resource allocation based on the cluster strategy to reduce the communication time. It is possible to plan a cloud

simulation workflow by using the experimental results, which demonstrate that the suggested approach outperforms the comparative algorithms without appreciably sacrificing individual fairness or overall time and cost.

In the paper [44] in 2022, to improve the ability of multi-objective evolutionary algorithms (MOEAs) to converge to non-dominated solutions, an improved multi-objective particle swarm optimization algorithm (IMaOPSO) is proposed by the authors, for dynamic scheduling of workflows in the cloud environments. The objective of the proposed approach is to address multi-objective QoS requirements in the context of cloud users and providers by minimizing makespan, cost and maximizing reliability for users, and minimizing energy consumption for providers. Addressing more than three objectives simultaneously in workflow planning is an aspect of innovation in this work that has not been addressed so far. MaOPSO uses a set of dynamically determining reference points based on the search process, allowing the algorithm to converge to non-dominated solutions.

Ye et al. [52] in 2022 presented an efficient stochastic multi-workflow dynamic scheduling algorithm called SMWDSA to schedule multi-workflows with deadline constraints for optimizing multi-workflow scheduling cost. The proposed SMWDSA consists of three stages including multi-workflows preprocessing, multi-workflow scheduling, and scheduling feedback. In SMWDSA, a novel task sub-deadlines assignment strategy is designed to assign the task sub-deadlines to each task of multi-workflow for meeting workflow deadline constraints. Then, they propose a task scheduling method based on the minimal time slot availability to execute tasks for minimizing workflow scheduling costs while meeting workflow deadlines. Finally, a scheduling feedback strategy is adopted to update the priorities and sub-deadlines of unscheduled tasks, for further minimizing workflow scheduling costs.

In 2022, Ye et al. [46] propose a workflow scheduling algorithm named REWS to reduce energy consumption and satisfy workflow reliability constraints. In REWS, a new sub-reliability constraint prediction strategy is adopted to break down the workflow reliability constraint into task sub-reliability constraints and the effectiveness of this strategy is proved. Moreover, an update method is adopted to adjust the task sub-reliability constraint for reducing energy consumption. In addition, a brief system framework which consists of five parts: workflow analyzer, reliability decomposer, resource manager, workflow scheduler, and feedback processer is built to support the algorithm implementation of REWS. They conduct the experiments using both synthetic data and real-world data to evaluate the proposed REWS approach.

Konjaang et al. [45] in 2022 developed a heuristic scheduling technique to solve the workflow task scheduling problem in a cloud computing environment. Moreover, workflow execution normally contains some time gaps (idle time), which may increase the execution cost of workflow tasks. This paper has proposed a slack time harvesting algorithm that is effective in making use of idle time. Finally, an energy-efficient VM Mapping Algorithm (EViMA) is proposed to support the scheduling model to achieve its aim of managing cloud resources to generate energy optimal schedules. EViMA is supported by four sub-algorithms including Highly Critical Workflow Task Selection Algorithm (HiCTSA), Low Critical Workflow Task Selection Algorithm (LoCTSA), VM Power Regulating Algorithm (VMPRA), and Slack Time Harvesting Method (STiHaM). EViMA produced the best solution that manages the use of cloud resources better, to reduce energy consumption, makespan, and execution cost.

In 2021, the paper [39] proposed an industrial workflow scheduling algorithm based on an adaptive genetic algorithm (GA) that utilizes multiple, software-defined cloud data center resources not only to improve energy utilization but also to keep the cost of execution to a minimum as well. First, a novel workflow broker based on SDNWB is used to deploy industrial workflow tasks across multiple software-defined data centers while automating task provisioning and data provisioning for this algorithm. Secondly, an adaptive genetic algorithm (GA) and SDNWB are combined for green planning of industrial workflow applications. The evaluation shows that this proposed method can increase the usage of green energy for the execution of industrial workflow up to three times with a slight increase of cost.

By determining the dynamic threshold value for scheduling jobs on virtual machines, [66] suggested a task's dynamic priority for workflow scheduling using MONWS, which applies the min-max algorithm to reduce finish time and maximize resource utilization. MONWS produced 35% improvements in makespan, 8% increases in maximum average cloud utilization, and 4% cost reductions when compared to existing algorithms based on the testing results.

5. Summary table of used algorithms and discussion

In this section, we will present the table of studied algorithms, and then we will analyze the table in the discussions.

5.1. Summary table of used algorithms

Ref	Authors	Algorithm	Wf Dy	RPS	Opt Stra	S O	Sim T
[54]	Zhu et al., 2015	EMO	Ensemble	Static	Meta-Heuristic	M, C	Amazon EC2
[31]	Rodriguez et al., 2017	BAGS	Single	Static	Heuristic	D, C	CloudSim
[32]	Singh et al., 2018	PDPPS	Single	Dynamic	Heuristic	D, C	Java
[33]	Arabnejad et al., 2018	BDAS	Single	Dynamic	Heuristic	M, C	CloudSim
[34]	Zhou et al., 2019	FDHEFT	Single	Static	Heuristic	M, C	JMetal
[35]	Gao et al., 2019	HGAABC	Single	Static	Hybrid	M, C	WorkflowSim
[5]	Liu et al., 2019	NOSF	Single	Dynamic	Heuristic	C, RU	Java
[36]	Faragardi et al., 2019	GRP-HEFT	Single	Dynamic	Heuristic	M, C	CloudSim
[37]	Kalyan et al., 2020	NBWS	Single	Static	Heuristic	M, C	CloudSim
[38]	Ndamlabin et al., 2020	CTTWS	Single	Static	Heuristic	D, C	WorkflowSim
[39]	Wen et al., 2020	GA	Single	Static	Meta-Heuristic	D, C, EC	CloudSim
[40]	Belgacem et al., 2022	HEFT-ACO	Single	Static	Hybrid	M, C	WorkflowSim
[41]	Ndamlabin et al., 2021	CTTWSDP	Single	Dynamic	Heuristic	D, C	WorkflowSim
[50]	Rajasekar et al., 2021	SRPSM	Multiple	Dynamic	Heuristic	D, C	CloudSim

[42]	Taghinezhad et al., 2021	BDDC + BDC	Single	Dynamic	Heuristic	D, C	Aneka
[55]	Arora et al., 2022	PSO–GWO	Single	Static	Hybrid	M, C	WorkflowSim
[56]	Konjaang et al., 2021	MOWOS	Single	Static	Heuristic	M, C	WorkflowSim
[43]	Hariri et al., 2022	MHPSLP	Single	Static	Heuristic	D, C	N/A
[51]	Feng et al., 2022	N/A	Multiple	Dynamic	Heuristic	M,C,F	WorkflowSim
[44]	Saeedi et al., 2020	IMaOPSO	Single	Static	Meta-Heuristic	M, C, R, EC	WorkflowSim
[52]	Ye et al., 2022	SMWDSA	Multiple	Dynamic	Heuristic	D, C	WorkflowSim
[46]	Ye et al., 2022	REWS	Single	Static	Heuristic	R, EC	Java
[45]	Konjaang et al., 2022	EViMA	Single	Dynamic	Heuristic	M, C, EC	WorkflowSim
[66]	Pillareddy et al., 2023	MONWS	Single	Dynamic	Heuristic	M, C	CloudSim

Table 1: Summarize Related Works

These columns in this table include the information below: Ref = Reference of article, RPS = Resource Provisioning Strategy, Wf Dy = Workflow Dynamicity, Opt Stra = Optimization Strategy (M = Makespan, C = Cost, D = Deadline, R = Reliability, RU = Resource Utilization, F = Fairness, EC = Energy Comsumption), SO = Scheduling Objectives, Sim T = Simulation Tool.

5.2. Discussion

In relation to the scheduling of scientific workflows in cloud computing IaaS infrastructures, the table 1 contains 24 papers classified from 2016 to 2023. These papers are classified according to some parameters, such as the dynamicity of scientific workflows, resource provisioning strategies, optimization strategies, scheduling objectives (metrics), and simulation tools.

- *Workflow dynamicity*

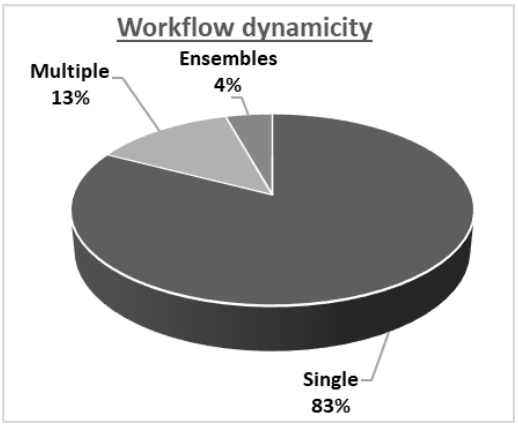

Fig. 7: Classification of Workflow Dynamicity.

Figure 7 shows the dynamicity of the workflows that arrive in a cloud system and have to be executed. Statistics show that 83% of the articles studied deal with a single workflow, 13% deal with multiple workflows and 4% deal with workflows ensembles. Among all these works selected between 2016 and 2022, we note that the majority of authors focus more on scheduling single workflows than multiple workflows and even less on scheduling workflow ensembles.

- *Resource provisioning strategy*

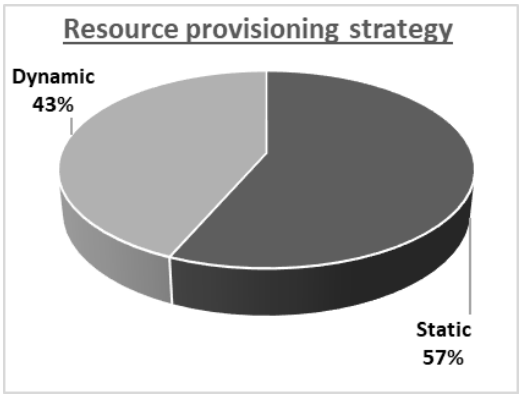

Fig. 8: Classification of Resource Provisioning Strategy.

Several studies demonstrate the importance of the resource provisioning strategy and these studies show that the resource provisioning strategy affects the scheduling objectives or metrics. Figure 8, shows the classification of the resource provisioning strategy. The strategy can be static (57%) or dynamic (43%) for the studies presented in the table. It is clear that when the resource provisioning strategy is static, the resources (VMs) are already reserved for the execution of the tasks of the different scientific workflows and can observe some

inactive time slots. For the dynamic resource provisioning strategy, resources are continuously provisioned as needed for the workflow tasks execution. It may happen in this case that the appropriate resources are not available at a given time. An ideal strategy must therefore be chosen that does not impact the optimization of metrics during workflow scheduling.

- *Optimization strategy*

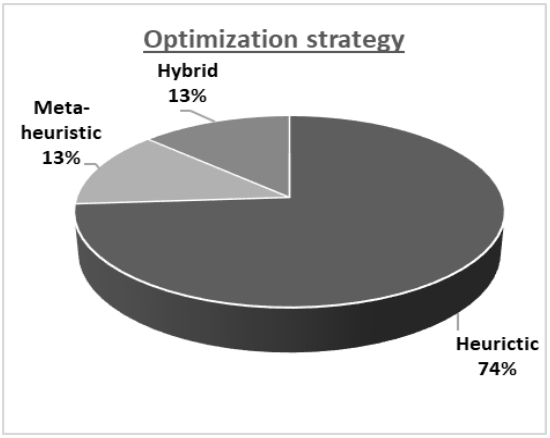

Fig. 9: Classification of Optimization Strategy.

For the optimization strategy presented in figure 8, we note that the majority of studies are done through heuristics (74%) and few studies use meta-heuristics (13%) and hybrid methods (13%). The heuristic strategies quickly find solutions to the problems at a lower cost and in a short time, contrary to the meta-heuristics and hybrid strategies. However, Meta-heuristics approaches produce better optimization results. Thus, hybrid (Heuristic / Meta-heuristic) approaches are likely recommended in dynamic environments like Cloud.

- *Scheduling objectives (metrics)*

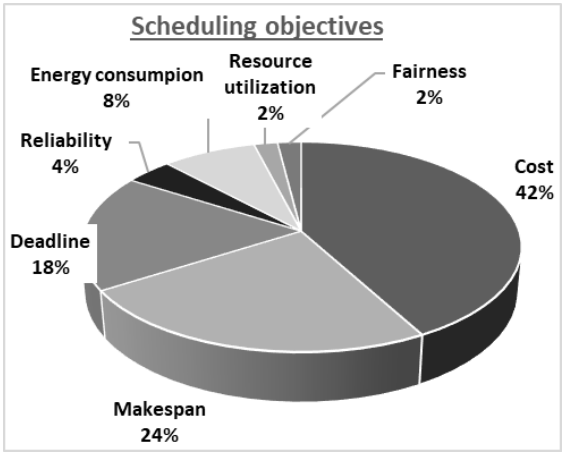

Fig. 10: Classification of Scheduling Objectives (Metrics) in the IaaS Cloud.

In the scheduling process of scientific workflows in cloud computing, the objective is to optimize some metrics which can be Cost, Makespan, Deadline, Reliability, Energy consumption, Resource utilization, and Fairness. Figure 10, shows a classification of the scheduling objectives (metrics) optimized in the different presented works. For the range of articles in the table, it is clear that the majority focus on metrics such as Cost (42%), Makespan (24%), and Deadline (18%), and few works focus on Reliability (4%), Energy consumption(8%), Resource utilization(2%), Fairness(2%). It should be noted that several works optimize two metrics at a time such as Makespan, Cost, and Deadline, Cost but few works optimize more than two metrics

- *Simulation tools*

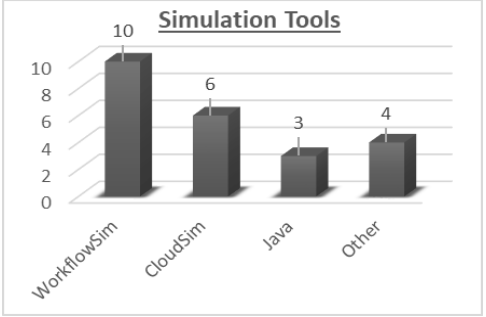

Fig. 11: Ranking of Simulation Tools.

Figure 11 gives a ranking of simulation tools for scientific workflows according to their usage preference.
It appears, from figure 11, that WorkflowSim is the most used tool by the authors and is as much used as all the other tools combined. CloudSim comes in second position in this ranking and is half as used as Workflowsim. It should be noted that Workflowsim is an extension of Cloudsim to simulate scientific workflows in distributed environments such as cloud computing.

6. Open issues

As future directions in the work of scheduling scientific workflows in IaaS infrastructures of cloud computing, we have a few points:

6.1. Scalability

Resources can't scale endlessly; thus, when a resource limit is reached, a service provider may choose to assign jobs to other providers in a transparent manner to consumers in order to minimize fines for SLA violations. A situation like this creates opportunities for SLA management studies.

6.2. Elasticity

In the context of cloud computing, elasticity refers to the dynamic placement and (re)sizing of an application's resources in accordance with the load that it is subjected to through addition, deletion, migration, and configuration changes. This problem is one of the main objectives and a major challenge of cloud computing.

6.3. Dynamic scheduling

It would be interesting to think about how the cloud environment is dynamic in order to choose the optimal resource allocation using continuously updated information.

6.4. Scaling up

By taking advantage of available computing power and scaling up in a cloud environment, it is possible to consider the parallelization of algorithms to increase their robustness and solve the multi-objective scheduling problem for large-scale workflows.

6.5. Communications between datacenters and a datacenter reputation

In a multi-datacenter cloud environment, workflows are submitted from any datacenter in the cloud. A workflow is submitted to the datacenter closest to its submission location. In order to comply with the Service Level Agreement (SLA) between each customer and the providers of the cloud resources, a datacenter may find itself obliged to transfer some or all of the tasks of a scientific workflow to a datacenter whose reputation for efficiency in executing a certain type of scientific workflow is known. It would be desirable to think of a distributed method where there is communication between datacenters in a cloud for the execution of scientific workflows. This will improve reliability and other optimization metrics.

6.6. Simulation environment

The majority of workflow scheduling techniques are implemented in cloud simulators like CloudSim and WorkflowSim. To see if the suggested methods work in practice, it will be best to simulate algorithms in real cloud computing environments.

7. Conclusion

Cloud computing swiftly evolved into a computer resource renting environment where users could also carry out their scientific operations. Finding effective scientific workflow scheduling algorithms that maximize a variety of metrics relating to QoS, the use of cloud resources, or environmental conservation is a challenge for scientists.
The study of algorithms for scheduling scientific workflows in a cloud computing environment is described in this paper. This study concentrated on workflow scheduling techniques, the way workflows arrive in the scheduler, the method for provisioning cloud resources to the scheduler, metrics for QoS, the use of cloud resources, or environmental protection factors taken into account by scheduling algorithms.
In this article, a taxonomy covering the study's subject is also offered. It is based on a thorough analysis of scientific workflow scheduling techniques. The taxonomy is accompanied by a categorization of the examined algorithms that exemplifies the most recent methods for the planning of scientific workflows in diverse aspects. The simulation tools for these scheduling techniques are likewise categorized. It was feasible to provide readers with an overview of the issue of scheduling scientific workflows in cloud computing and with fresh directions for future research thanks to the analysis of these various classifications.

References

[1] R. Duan, T Fahringer, R. Prodan, J. Qin, A. Villazon, and M. Wieczorek. Real world workflow applications in the askalon grid environment. European Grid Conference, Springer, page 454–463, 2005. https://doi.org/10.1007/11508380_47.
[2] X. Geng, Y. Mao, M. Xiong, and Y. Liu. An improved task scheduling algorithm for scientific workflow in cloud computing environment. Cluster Computing, 22:7539–7548, 2019. https://doi.org/10.1007/s10586-018-1856-1.
[3] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. A performance analysis of ec2 cloud computing services for scientific computing. International Conference on Cloud Computing, 2009. https://doi.org/10.1007/978-3-642-12636-9_9.
[4] C. N. Hoefer and G. Karagiannis. Taxonomy of cloud computing services. EEE Globecom Workshops, page 1345–1350, 2010. https://doi.org/10.1109/GLOCOMW.2010.5700157.

[5] J. Liu, J. Ren, W. Dai, D. Zhang, P. Zhou, Y. Zhang, G. Min, and N. Najjari. Online multi-workflow scheduling under uncertain task execution time in iaas clouds. IEEE Transactions on Cloud Computing, 2019.

[6] M. Zhu, Q. Wu, and Y. Zhao. A cost-effective scheduling algorithm for scientific workflows in clouds. IEEE 31st International Performance Computing and Communications Conference (IPCCC), page 256–265, 2012.

[7] S. J. Nirmala and S. M. S. Bhanu. Catfish-pso based scheduling of scientific workflows in iaas cloud. Computing 98, page 1091–1109, 2016. https://doi.org/10.1007/s00607-016-0494-9.

[8] W. Jing, W. Qiang, and L. Xiongfei. A data placement and task scheduling algorithm in cloud computing. Journal of Computer Research and Development, 51, 2014.

[9] R. Ghafouri, A. Movaghar, and M. Mohsenzadeh. A budget constrained scheduling algorithm for executing workflow application in infrastructure as a service clouds. Peer-to-Peer Networking and Applications, 12:241–268, 2018. https://doi.org/10.1007/s12083-018-0662-0.

[10] J. Sun, L. Yin, M. Zou, Y. Zhang, T. Zhang, and J. Zhou. Makespan minimization workflow scheduling for complex networks with social groups in edge computing. Journal of Systems Architecture, 2020. https://doi.org/10.1016/j.sysarc.2020.101799.

[11] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang. Cost-efficient task scheduling for executing large programs in the cloud. Parallel Computing, 39:177–188, 2013. https://doi.org/10.1016/j.parco.2013.03.002.

[12] V. Singh, I. Gupta, and P. K. Jana. An energy efficient algorithm for workflow scheduling in iaas cloud. Journal of Grid Computing, 18:357–376, 2020. https://doi.org/10.1007/s10723-019-09490-2.

[13] X. Liu, W. Dou, J. Chen, S. Fan, S.C. Cheung, and S. Cai. On design, verification, and dynamic modification of the problem-based scientific workflow model. Simulation Modelling Practice and Theory, 15:1068–1088, 2007. https://doi.org/10.1016/j.simpat.2007.06.003.

[14] A. Asghari, M. K. Sohrabi, and F. Yaghmaee. Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel sarsa reinforcement learning agents and genetic algorithm. The Journal of Supercomputing, page 1–29, 2020. https://doi.org/10.1007/s11227-020-03364-1.

[15] X. Ye, J. Liang, S. Liu, and J. Li. A survey on scheduling workflows in cloud environment. International Conference on Network and Information Systems for Computers, page 344–348, 2015. https://doi.org/10.1109/ICNISC.2015.91.

[16] Vijindra and S. Shenai. Survey on scheduling issues in cloud computing. Procedia Engineering, 38:2881–2888, 2012. https://doi.org/10.1016/j.proeng.2012.06.337.

[17] K. Setrag and B. Marek. Groupware and workflow. Paris : Masson, 1998.

[18] P. Lawrence. Workflow management coalition. Workflow Handbook, 1997.

[19] T. E. El-Sayed, I.E Ali, and F.A Mohamed. Extended max-min scheduling using petri net and load balancing. Int. J. Soft Comput. Eng. (IJSCE), 2:198–203, 2012.

[20] I. Gupta, S. Gupta, A. Choudhary, and Jana P.K. A hybrid meta-heuristic approach for load balanced workflow scheduling in iaas cloud. Lecture Notes in Computer Science, 11319:73–89, 2019. https://doi.org/10.1007/978-3-030-05366-6_6.

[21] J. Gideon, C. Ann, D. Ewa, B. Shishir, M. Gaurang, and V. Karan. Characterizing and profiling scientific workflows. Future Generation Computer Systems, 29:682–692, 2013. https://doi.org/10.1016/j.future.2012.08.015.

[22] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, J. Lee, E.A.and Tao, and Y Zhao. Scientific workflow management and the kepler system. Concurrency and computation: Practice and experience, 18:1039–1065, 2006. https://doi.org/10.1002/cpe.994.

[23] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, A. Wipat, and P.L. Taverna. Lessons in creating a workflow environment for the life sciences. GGF10, 2004. https://doi.org/10.1002/cpe.993.

[24] I. Taylor, M. Shields, I. Wang, and A. Harrison. The triana workflow environment: Architecture and applications. In Workflows for e-Science, Springer, pages 320–339, 2007. https://doi.org/10.1007/978-1-84628-757-2_20.

[25] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto Jr, and H.L. Truong. Askalon: a tool set for cluster and grid computing. Concurrency and Computation: Practice and Experience, 17:143–169, 2005. https://doi.org/10.1002/cpe.929.

[26] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, and A. Laity. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming, 13:219–237, 2005. https://doi.org/10.1155/2005/128026.

[27] G.J. Toomer. Ptolemy's almagest. Princeton: Princeton University Press, 712, 1998. https://doi.org/10.1515/9780691213361.

[28] Y.H. Jia, W.N. Chen, H. Yuan, T. Gu, H. Zhang, Y. Gao, and J. Zhang. An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51:634–649, 2018. https://doi.org/10.1109/TSMC.2018.2881018.

[29] S. Yassa, R. Chelouah, H. Kadima, and B. Granado. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. The Scientific World Journal, pages 634–649, 2013. https://doi.org/10.1155/2013/350934.

[30] S. Yassa, J. Sublime, R. Chelouah, H. Kadima, G.S. Jo, and B. Granado. A genetic algorithm for multi-objective optimisation in workflow scheduling with hard constraints. International Journal of Metaheuristics, 2:415–433, 2013. https://doi.org/10.1504/IJMHEUR.2013.058475.

[31] M.A. Rodriguez and R. Buyya. Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 12:1– 22, 2017. https://doi.org/10.1145/3041036.

[32] V. Singh, I. Gupta, and P.K. Jana. A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. Future Generation Computer Systems, 79:95–110, 2018. https://doi.org/10.1016/j.future.2017.09.054.

[33] V. Arabnejad, K. Bubendorfer, and B. Ng. Budget and deadline aware e-science workflow scheduling in clouds. IEEE Transactions on Parallel and Distributed systems, 30:29–44, 2018. https://doi.org/10.1109/TPDS.2018.2849396.

[34] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft. Future Generation Computer Systems, 93:278–289, 2019. https://doi.org/10.1016/j.future.2018.10.046.

[35] Y. Gao, S. Zhang, and J. Zhou. A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud. IEEE Access, 7:125783–125795, 2019. https://doi.org/10.1109/ACCESS.2019.2939294.

[36] H.R. Faragardi, M.R.S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli. Grp-heft: A budget-constrained resource provisioning scheme for workflow scheduling in iaas clouds. IEEE Transactions on Parallel and Distributed Systems, 31:1239–1254, 2019. https://doi.org/10.1109/TPDS.2019.2961098.

[37] K. Kalyan Chakravarthi, L. Shyamala, and V. Vaidehi. Budget-aware scheduling algorithm for workflow applications in iaas clouds. Cluster Computing, 23:3405–3419, 2020. https://doi.org/10.1007/s10586-020-03095-1.

[38] J.E. Ndamlabin Mboula, V.C. Kamla, and C.T. Djamegni. Cost-time trade-off efficient workflow scheduling in cloud. Simulation Modelling Practice and Theory, 103:102107, 2020. https://doi.org/10.1016/j.simpat.2020.102107.

[39] Z. Wen, S. Garg, G.S. Aujla, K. Alwasel, D. Puthal, S. Dustdar, A.Y. Zomaya, and R. Ranjan. Running industrial workflow applications in a software-defined multicloud environment using green energy aware scheduling algorithm. IEEE Transactions on Industrial Informatics, 17:5645–5656, 2020. https://doi.org/10.1109/TII.2020.3045690.

[40] A. Belgacem and K. Beghdad-Bey. Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. Cluster Computing, 25:579–595, 2022. https://doi.org/10.1007/s10586-021-03432-y.

[41] J.E. Ndamlabin Mboula, V.C. Kamla, and C. Tayou Djam´egni. Dynamic provisioning with structure inspired selection and limitation of vms based cost-time efficient workflow scheduling in the cloud. Cluster Computing, 24:2697–2721, 2021. https://doi.org/10.1007/s10586-021-03289-1.

[42] A. Taghinezhad-Niar, S. Pashazadeh, and J. Taheri. Workflow scheduling of scientific workflows under simultaneous deadline and budget constraints. Cluster Computing, 24:3449–3467, 2021. https://doi.org/10.1007/s10586-021-03314-3.

[43] M. Hariri, M. Nouri-Baygi, and S. Abrishami. A hybrid algorithm for scheduling scientific workflows in iaas cloud with deadline constraint. The Journal of Supercomputing, pages 1–22, 2022. https://doi.org/10.1007/s11227-022-04563-8.

[44] S. Saeedi, R. Khorsand, S.G. Bidgoli, and M. Ramezanpour. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. Computers and Industrial Engineering, 147:106649, 2020. https://doi.org/10.1016/j.cie.2020.106649.
[45] J.K. Konjaang, J. Murphy, and L. Murphy. Energy-efficient virtual-machine mapping algorithm (evima) for workflow tasks with deadlines in a cloud environment. Journal of Network and Computer Applications, 203:103400, 2022. https://doi.org/10.1016/j.jnca.2022.103400.
[46] L. Ye, Y. Xia, S. Tao, C. Yan, R. Gao, and Y. Zhan. Reliability-aware and energy-efficient workflow scheduling in iaas clouds. IEEE Transactions on Automation Science and Engineering, 2022. https://doi.org/10.1109/TASE.2022.3195958.
[47] M. Marwa, J.E. Hajlaoui, Y. Sonia, M.N. Omri, and C. Rachid. Multi-agent system-based fuzzy constraints offer negotiation of workflow scheduling in fog-cloud environment. Computing, pages 1–33, 2023. https://doi.org/10.1007/s00607-022-01148-4.
[48] M. Mokni, S. Yassa, J.E. Hajlaoui, M.N. Omri, and R. Chelouah. Multi-objective fuzzy approach to scheduling and offloading workflow tasks in fog–cloud computing. Simulation Modelling Practice and Theory, 123:102687, 2023. https://doi.org/10.1016/j.simpat.2022.102687.
[49] M.A. Rodriguez and R. Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. Concurrency and Computation: Practice and Experience, 29:4041, 2017. https://doi.org/10.1002/cpe.4041.
[50] P. Rajasekar and Y. Palanichamy. Scheduling multiple scientific workflows using containers on iaas cloud. Journal of Ambient Intelligence and Humanized Computing, 12:7621–7636, 2021. https://doi.org/10.1007/s12652-020-02483-0.
[51] F. Li and W. Jun. Multi-objective optimization of clustering-based scheduling for multi-workflow on clouds considering fairness. arXiv preprint arXiv:2205.11173, 2022. https://doi.org/10.2139/ssrn.4128847.
[52] L. Ye, Y. Xia, L. Yang, and Y. Zhan. Dynamic scheduling stochastic multiworkflows with deadline constraints in clouds. IEEE Transactions on Automation Science and Engineering, 2022. https://doi.org/10.1109/TASE.2022.3204313.
[53] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski. Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. Future Generation Computer Systems, 48:1–18, 2015. https://doi.org/10.1016/j.future.2015.01.004.
[54] Z. Zhu, G. Zhang, M. Li, and X. Liu. Evolutionary multi-objective workflow scheduling in cloud. IEEE Transactions on parallel and distributed Systems, 27:1344–1357, 2015. https://doi.org/10.1109/TPDS.2015.2446459.
[55] N. Arora and R.K. Banyal. A particle grey wolf hybrid algorithm for workflow scheduling in cloud computing. Wireless Personal Communications, 122:3313–3345, 2022. https://doi.org/10.1007/s11277-021-09065-z.
[56] J.K. Konjaang and L. Xu. Multi-objective workflow optimization strategy (mowos) for cloud computing. Journal of Cloud Computing, 10:1–19, 2021. https://doi.org/10.1186/s13677-020-00219-1.
[57] EG. Talbi. Metaheuristics: from design to implementation. John Wiley and Sons, 2009. https://doi.org/10.1002/9780470496916.
[58] Sonia Yassa. Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing. Th`ese de doctorat, Universit´ de Cergy Pontoise, 2014.
[59] N.A.B. Mary and K. Jayapriya. An extensive survey on qos in cloud computing. International Journal of Computer Science and Information Technologies, 5:1–5, 2014.
[60] K. Liu, H. Jin, J. Chen, X. Liu, D. Yuan, and Y. Yang. An extensive survey on qos in cloud compa compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. The International Journal of High-Performance Computing Applications, 24:445–456, 2010. https://doi.org/10.1177/1094342010369114.
[61] J. Bushra, Humaira I., Mohammad S., Kashif M., and Rajkumar B. Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. ACM Computing Surveys, 10, 2022.
[62] M.A. Rodriguez and R. Buyya. Deadline-based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE transactions on cloud computing, 2:222–235, 2014. https://doi.org/10.1109/TCC.2014.2314655.
[63] L. Zhao, Y. Ren, and K. Sakurai. Reliable workflow scheduling with less resource redundancy. Parallel Computing, 39:567–585, 2013. https://doi.org/10.1016/j.parco.2013.06.003.
[64] Q. Tao, H. Chang, Y. Yi, C. Gu, and Y. Yu. Qos constrained grid workflow scheduling optimization based on a novel pso algorithm. Eighth International Conference on Grid and Cooperative Computing, pages 153–159, 2009. https://doi.org/10.1109/GCC.2009.39.
[65] M.A. Rodriguez and R. Buyya. A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds.44th International Conference on Parallel Processing, pages 839–848, 2015. https://doi.org/10.1109/ICPP.2015.93.
[66] V.R. Pillareddy and G.R. Karri. Monws: Multi-objective normalization workflow scheduling for cloud computing. Applied Sciences, 13:1101, 2023. https://doi.org/10.3390/app13021101.