Cluster Computing
https://doi.org/10.1007/s10586-021-03432-y

Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost

Ali Belgacem[1] · Kadda Beghdad-Bey[2]

Received: 20 March 2021 / Revised: 1 July 2021 / Accepted: 24 September 2021

Abstract
Recently, modern businesses have started to transform into cloud computing platforms to deploy their workflow applications. However, scheduling workflow under resource allocation is significantly challenging due to the computational intensity of the workflow, the dependency between tasks, and the heterogeneity of cloud resources. During resource allocation, the cloud computing environment may encounter considerable problems in terms of execution time and execution cost, which may lead to disruptions in service quality given to users. Therefore, there is a necessity to reduce the makespan and the cost at the same time. Often, this is modeled as a multi-objective optimization problem. In this respect, the fundamental research issue we address in this paper is the potential trade-off between the makespan and the cost of virtual machine usage. We propose a HEFT-ACO approach, which is based on the heterogeneous earliest end time (HEFT), and the ant colony algorithm (ACO) to minimize them. Experimental simulations are performed on three types of real-world science workflows and take into account the properties of the Amazon EC2 cloud platform. The experimental results show that the proposed algorithm performs better than basic ACO, PEFT-ACO, and FR-MOS.

Keywords Cloud computing · Workflow scheduling · Resource allocation · Makespan · Cost · ACO algorithm

1 Introduction

Information technology (IT) and the Internet have become essential elements for human life. However, in the last years, the IT sector became very large and complex, which necessitated migrating users from traditional software models to cloud computing. Cloud is a popular IT model that can afford to process large volumes of data. It stems from a long history of research and takes grid computing as a basic structure. Today, cloud computing holds a primordial place in distributed computing. Indeed, it allows access to large amounts of computing power in a fully virtualized manner by aggregating resources and offering a single system view [1].

Virtualization is a necessary technique to make the cloud more automatic and agile. It allows returning a physical operating system, the computing device (server), storage device, or network devices to virtual devices. In other words, virtualization is the transparent emulation of an IT resource-producing to its consumers' benefits that were unavailable in its physical form [2]. Datacenter virtualization is a primordial process to make the cloud more efficient and scalable. Whereas virtual machines (VMs) are the fundamental components of the cloud. They are formed according to the software layer added to a real machine. By doing so, VM can circumvent the real machine compatibility and hardware resource constraints [3].

The cloud is a large pool of shared resources with free access in a dynamic, scalable manner and with guaranteed quality of service [4–6]. It aims to provide users with resources for an agreed period of time and satisfy both the needs of users and service providers. The strong evolution of the Cloud and the migration of many network services to such an elastic environment make it a crucial research subject. Resource allocation is one of the important research trends that must be targeted to maintain quality of

✉ Ali Belgacem
a.belgacem@univ-boumerdes.dz

Kadda Beghdad-Bey
k.beghdadbey@gmail.com

[1] M'hamed Bougara University, Boumerdes, Algeria

[2] École Militaire Polytechnique, Algiers, Algeria

Published online: 10 October 2021

service (QoS). Formally, this problem involves a set of tasks planned to execute on a set of resources. While this operation is subjected to a set of constraints to optimize an objective function. Therefore, the ultimate goal is to build a plan which specifies when and on what resource each task will be executed. In general, there are no algorithms that can produce an optimal solution in polynomial time for these kinds of problems [7]. Moreover, since the solutions are based on exhaustive research that is very expensive and restrictive, the allocation of resources in a cloud environment belongs to a category of problems called NP-hard.

On the other hand, workflow software application technology continues to be subjected to on-going growth in business and science areas. It is designed as a set of automating depending tasks. Its design can involve considerations such as the type of tasks involved, their interaction, and the sequence in which they need to be completed (i.e. sequential or parallel) [8]. Workflows are divided into business and scientific categories. The first is a group of operations to perform a task comprising at least two human factors. While the second is used in medicine, meteorology, etc. Workflows differ based on their type (business or scientific) and type of dependencies among their tasks [9]. Indeed, cloud providers have often designed and hosted workflow platform services. Furthermore, the workflow scheduling process is subjected to two main strategies. The first is the ready task selection phase. During this stage, the tasks that can be scheduled at the same time without dependent constraints are extracted using different techniques. In the second stage, the schedule is generated, and each task is mapped onto the best-suited resource [10].

Resources management is considered an important aspect to obtain the desired objective behind resource allocation. This issue is solved using techniques based on metaheuristics. They allow providing almost optimal solutions within a reasonable time. Many researchers have approached these techniques while trying to improve certain points and satisfy different metrics. Particularly and in the context of dependent tasks, metaheuristics have many advantages which facilitate the processing of a workflow. The ant colony is widely used due to its good results in solving different combinatorial problems. It is considered one of the most powerful metaheuristics [11]. Its strength comes from the fact that it adopts a constructive method to obtain better results.

Our contribution in this study is to propose an algorithm for workflow scheduling in cloud infrastructure. The proposed approach is developed on the basis of an ant colony algorithm aimed to minimize the makespan and cost. The rest of the paper is organized as follows: Sect. 2 presents related work of workflow scheduling. The studied problem is described and formulated in Sect. 3. In Sect. 4 we introduce the proposed approach processes. Section 5 presents an experimental evaluation of the performance of our heuristic. Section 6 discusses the advantages and limits of this work. Section 7 concludes the paper.

2 Related work

The workflow scheduling problem is an important topic to research in the cloud computing area. It is characterized by its task dependencies which make researchers solve it in different ways according to the targeted objective (Table 1). For example, in [12], researchers proposed a heuristic approach that combines the modified analytic hierarchy process (MAHP), bandwidth aware divisible scheduling (BATS) + BAR optimization, longest expected processing time preemption (LEPT), and divide-and-conquer to perform task scheduling and resource allocation. This approach gives accepted results in terms of turnaround time and response time. In the paper [13], they deal with the challenges of scheduling workflows to improve both makespan and robustness, using the Dynamic Earliest-Finish-Time (DEFT) algorithm. To our knowledge, this algorithm needs to be further improved to be more suited to failure tasks. A new heuristic scheduling algorithm called Budget Deadline Aware Scheduling (BDAS) is introduced in work [14]. It addressed the scheduling of the scientific workflow in accordance with budget constraints and deadlines. BDAS significantly reduces the workload on instances.

To improve the reliability of scientific workflow execution, researchers at [15] proposed a Fault Tolerant Workflow (DFTWS) scheduling approach with hybrid spatial and temporal re-execution schemes. The main advantage of DFTWS is to make a compromise between high reliability and low-cost objectives. However, the workflow is not flexibly scheduled. In [16], they designed a new algorithm named FDHEFT for the workflow scheduling problem. Its goal is to minimize costs and makespan simultaneously under the precedence constraints of workflow tasks. FDHEFT can achieve a significantly better cost-makespan trade-off with remarkably higher Hypervolume and can run up to hundreds of times faster than existing algorithms. However, this solution did not take into account the overhead time related to communication between the nodes. In work [9], the authors proposed a model for task scheduling and fragmentation in scientific workflows. They solved the problem studied in terms of bandwidth and memory costs by taking into account the runtime conditions and the number of VMs without considering the heterogeneity of resources.

A new solution for workflow scheduling based on a multi-objective genetic algorithm is proposed in work [17].

Table 1 Treating the dependency of tasks constraint in workflow

Works	Dependency of tasks for workflow scheduling
[36]	Go through the workflow and find the parallel tasks in order to schedule them at the same time
[37]	Critical paths (critical path first, dynamic critical path, critical-path-on-a-processor)
[38]	The workflow is divided into several levels that can be executed sequentially, and tasks within one level do not do depend on each other. Thereby the tasks are assembled in clusters.
[39]	Assemble the ready tasks of each workflow into a priority queue according to the priority of the workflow to which they belong.
[22]	Using a first-come, first-served (FCFS) queue to schedule the workflow
[21]	Using a plain upward rank, where each task is represented by the accumulated processing time of successors on its critical path and its weight (the stationary probability) which gives the importance of the task in the global DAG topology.
[9]	Critical path lookup
[20]	Horizontal, vertical, and diagonal fragmentation of workflow
[16]	Task prioritizing using HEFT algorithm

It addressed the conflicting interests of cloud stakeholders, such as makespan, execution costs, power consumption, and resource utilization regardless of the communication time between tasks. To our knowledge, this solution needs to be further improved to schedule the workflow. In order to minimize the total execution price (TPE) and total execution time (TET) of the workflow, the authors of work [18] proposed an S-CEDA strategy. S-CEDA is implemented to meet the deadline constraints in a stochastic environment. However, this solution needs to be compared to other existing work to prove its effectiveness. Similarly, in work [19], the researchers implemented the Jaya optimization algorithm to minimize the cost of execution and makespan with neglecting the deadline.

In [20], the authors focused on investigating a dynamic scheduling method for concurrent workflows named DSM. DSM aims to reduce the time overhead and maximize resource utilization. However, other necessary factors must be taken into account, including transmission time and resource characteristics. The work [21] concedes the budget constraint factor to schedule a complex workflow. This issue is formulated as an integer programming problem, which uses the Markovian chain stationary probabilities to measure the properties of the workflow tasks. However, the authors did not use a deep analysis of the structural characteristics of different workflows. In work [22], the researchers proposed a new approach for multi-purpose workflow scheduling that aims to optimize the workflow makespan and cost simultaneously. This work did not consider the data transmission cost.

Due to the increasing complexity of the workflow application, researchers in [23] used the Predicted Earliest finish time (PEFT) with the ACO algorithm to solve it. This approach allows for reducing costs and saving energy consumption. However, it showed poor performance in terms of execution time for a high workload. Moreover, in [24], they used two scheduling algorithms Ant Colony Optimization and Black Hole Algorithm, to reduce the makespan and total execution cost to the user. However, this solution needs more experiments to prove its effectiveness. The multi-objective scheduling algorithm with fuzzy resource utilization (FR-MOS) proposed in [25] aimed to minimize cost and makespan considering the reliability constraints. FR-MOS gives a better performance in terms of success rate and monetary cost. In order to minimize execution cost under a user-defined deadline, the authors of [26] propose two list scheduling algorithms named Look-back Workflow Scheduling (LBWS) and Structure AwareWorkflow scheduling (SAWS). However, both works are shown unsatisfactory results in terms of makespan. In paper [17], the authors addressed the conflicts that may be between makespan, execution cost, energy consumption, and resource utilization. While in [27], they proposed a Firefly algorithm (FA) to deal with multiple conflicting objectives including, the workload of cloud servers, makespan, resource utilization, and reliability. The researchers treated different objectives at the same time, which affect worse in the quality of results. Also, these works still need more evaluation to prove their effectiveness for resource allocation in the cloud.

There are recent works that apply new methods belongs to a modern branch of science, such as deep learning and decision-making techniques to optimize the scientific research problems. For example, a new prediction method presented in [28] aimed to reduce air pollution. It is used deep analysis and dataset for intelligent computing of the error between actual and predicted values. In the same trend, a dynamic mathematical model is proposed in [29]. It combines data mining algorithms and Fuzzy All Permutation Rule Base (FARB) for intelligent data analysis for huge and small datasets. In addition, a statistical tool generating the key of the stream cipher system was

improved in the paper [30]. Likewise, in [31], the researchers proposed a smart data analysis model to schedule the activities of smartphones and smartwatches. It uses a Random Forest-Data Mining (RF-DM) algorithm for Key generator.

Another method named developed random forest and local least squares (DRFLLS) is presented in [32]. DRFLLS aimed to estimate missing values of various datasets. While the multi-objective method presented in the article [33], focused on improving the production of electrical energy from wind. Similarly, a predicted tool based on developing MARS data mining technique is introduced in [34] to solve a multi-objectives optimization problem. In the same vein, paper [35] gave a new nature-inspired algorithm to find the optimal proteins generated through DNA synthesis. However, these recent algorithms have not been tested to schedule workflow under resource allocation in cloud computing environments.

3 Cloud workflow scheduling formulation

The proposed HEFT-ACO approach is designed to improve QoS requirements, namely, makespan and cost. We consider the cloud system model consists of three layers named, the cloud user, the workflow scheduler, and the cloud resource, as illustrated in Fig. 1. The first layer consists of a set of the cloud user, who submits the workflow tasks to the scheduler. The second layer contains the scheduling algorithm. The last layer of the proposed model is the cloud resource (Infrastructure as a Service (IaaS)). The cloud user submits the workflow to the waiting list, which operates on a first-come-first-served basis and according to the available virtual machines. The workflows are sent to the scheduling algorithm, which is used to find the optimal solution for two contradictory objectives the makespan and the cost. Then the resources are allocated according to the plan generated by the algorithm. After that, the corresponding user is served. The main symbols used in this study are summarized in Table 2.

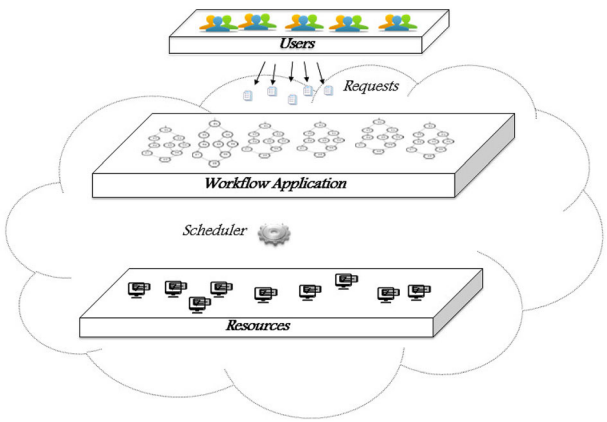


Fig. 1 Problem description

3.1 Workflow application model

The workflow application is modeled as a Directed Acyclic Graph (*DAG*). *DAG* represents the relationship between the tasks of the workflow graph (*WG*). More precisely, $WG = (E, T, Dt)$ is structured by a set of directed edges $E = \{e_1, e_2, \ldots, e_h\}$ defining the dependency between a set of tasks $T = \{T_1, T_2, \ldots, T_m\}$. The e_{ij} represents a precedence constraint that the task T_i cannot start its execution before T_j completes and sends all the needed output data (Dt) to the task T_i. Therefore, a task T_i may have multiple predecessors (Pr) and multiple successors (Sc) tasks defined as $Pr(T_i)$ and $Sc(T_i)$, respectively. For a given WG, its entry task is noted as T_0, where $Pr(T_0) = \emptyset$. Also, its output task is noted as T_{end} where $\exists T_i \in T : T_{end} \in (T_i) \vee Sc(T_{end}) = \emptyset$ and $Dt = \{T_i, Pr(T_i) | T_i \in T\}$.

3.2 Cloud resource manage model

The cloud environment adopted in this study is similar to that of Amazon Elastic Compute Cloud (EC2) used to perform workflow tasks. EC2 provides complete control of the computing resources and allows users to choose different processors, RAM, and bandwidth configurations. The EC2 instance (running a virtual machine) can be equipped with a temporary instance store to provide sufficient space for the data files resulting from the execution of the tasks [40]. In this way, the EC2 platform easily manages the large interconnection between tasks.

Based on the foregoing, our hardware platform is represented by a set of heterogeneous virtual machines (*VMs*). They can be of any type as provided by an IaaS provider. The virtual machines are configured with different parameters of resources. Moreover, they can be placed in different physical machines (Hosts), the average bandwidth between them denoted BW (it is assumed different according to VM type). Each VM_r from the set $VM = \{VM_1, VM_2, \ldots, VM_n\}$ is characterized by its own CPU and memory configuration. Where the CPU performance is presented by the parameter "Million Instructions Per Second (*MIPS*)".

3.3 Time model

This study takes into consideration different time types due to the dependence constraint between tasks, as shown in the following:

Table 2 Symbols

Symbols	Signification	Symbols	Signification
Pr	Predecessors	φ	Execution time
Sc	Successor	W	Task weight
Dt	Output data	φ	Execution time
WG	Workflow graph	$D\varphi$	Data transfer time
BW	Bandwidth	$S\varphi$	Start time
m	Tasks number	$F\varphi$	Finish time
n	VMs number	ϕ	Execution cost
h	Precedent edges number	Cw	Communication weight
r	VMs order	j	Predecessors of T_i order
ph	The pheromone value	$E\varphi$	Eearliest execution time estimation
$E\phi$	Cost estimation	Cd	Crowding distance

3.3.1 Execution time (φ)

The execution time of a given task in the workflow is defined as the time spent by a virtual machine to complete that task. It is expressed as a relationship shown in Eq. 1.

$$\varphi_{ir} = \frac{W_i}{MIPS_r} \tag{1}$$

where W is the weight (length) of the task.

3.3.2 The data transfer time ($D\varphi$)

The data transfer between tasks is calculated using the following equation:

$$D\varphi_{ij} = \begin{cases} 0 & \text{if } T_i \text{ and } T_j \text{ are executed on the same machine} \\ \dfrac{T_{file_i}}{BW_j} & \text{otherwise} \end{cases} \tag{2}$$

where T_{file} is the data transferred through the edge e_{ij}.

3.3.3 Start time ($S\varphi$)

It corresponds to the maximum execution time of the last task among the previous tasks of T_i and their sum of data transfer time (Eq. 3).

$$S\varphi_i = Max\{\varphi(T_j)\} + \sum_{j=0}^{h} D\varphi_{ij}, \quad where \quad T_j \in Pr(T_i) \tag{3}$$

3.3.4 Finish time ($F\varphi$)

It is the relationship between execution time and the start time of task (T_i)(Eq. 4).

$$F\varphi_i = \varphi_{ir} + S\varphi_i \tag{4}$$

3.3.5 Makespan

The overall time to complete workflow is called a makespan. It is defined as the maximum duration of execution of all workflow tasks (Eq. 5).

$$makespan = Max(F\varphi) \tag{5}$$

3.4 Cost model

The cost of scheduling workflow is computing according to the following:

3.4.1 Task execution cost (ϕ)

The executing cost of a given task is calculated by multiplying the total time spent running the task on the selected virtual machine by the price of that VM (Eq. 6).

$$\phi_{ir} = C_r * (F\varphi_i - S\varphi_i + I_r) \tag{6}$$

where C_r is the price of using the VM_r. While I is the virtual machine preparation time.

3.4.2 Workflow execution cost

The overall cost of running workflow is the sum of the execution cost of all its tasks (Eq. 7).

$$Cost = \sum_{i=0}^{m} \phi_i \tag{7}$$

3.5 Objective function

The objective function aim is to minimize the overall completion time and the overall cost of task executions.

$$\theta = Min(makespan, cost) \tag{8}$$

Subject to:

$$F\varphi_{ij} + S\varphi_i + I_r > \sum F\varphi_j \qquad \forall j \in Pr(T_i) \tag{9a}$$

$$\sum_{i=0}^{n} T_i = 1 \tag{9b}$$

Constraint 8a ensures that the task does not run only after its previous tasks have been completed and all required inputs have been received. While constraint 8b means each task is assigned to one virtual machine.

4 Proposed workflow scheduling approach

This work focuses on the application of a metaheuristic algorithm to find a better solution that assigns workflow tasks to VMs. The workflow schedule is doing mainly according to two phases. The first aimed to rank the dependent tasks. While the second is used to schedule ready tasks. The flowchart of the proposed approach is given in Fig. 2.

4.1 Workflow tasks ranking phase

To determine the execution tasks rank and to avoid the random execution tasks rank at the same level, the technique of weighting is used. This technique is based on the upward rank heuristic and is used on acyclic graphs directed by the HEFT algorithm (heterogeneous earliest finish time) [41]. Since the task weight rank can affect the performance of our algorithm, for each task, it is calculated by the formula:

$$Rank_i = \bar{\varphi}_i + max(Cw_{ij} + rank(T_j)), \quad T_j \in Sc(T_i) \tag{10}$$

$$\bar{\varphi}_i = \frac{1}{n}\sum_{r=0}^{n} \varphi_{ir} \tag{11a}$$

$$Cw_{ij} = \frac{\sum D\varphi_{ij}}{\sum Pr(T_j)} \tag{11b}$$

where Cw_{ij} is the communication weight via the edges e_{ij}. $\bar{\varphi}_i$ is the average execution time of the task T_i on all the VM_r. In this phase, the majority of the parameters are initialized in order to perform the calculation of the weights and maintain a robust order of the tasks (Fig. 3). Other main results of this phase are execution time φ_{ir} matrix and cost matrix ϕ_{ir}.

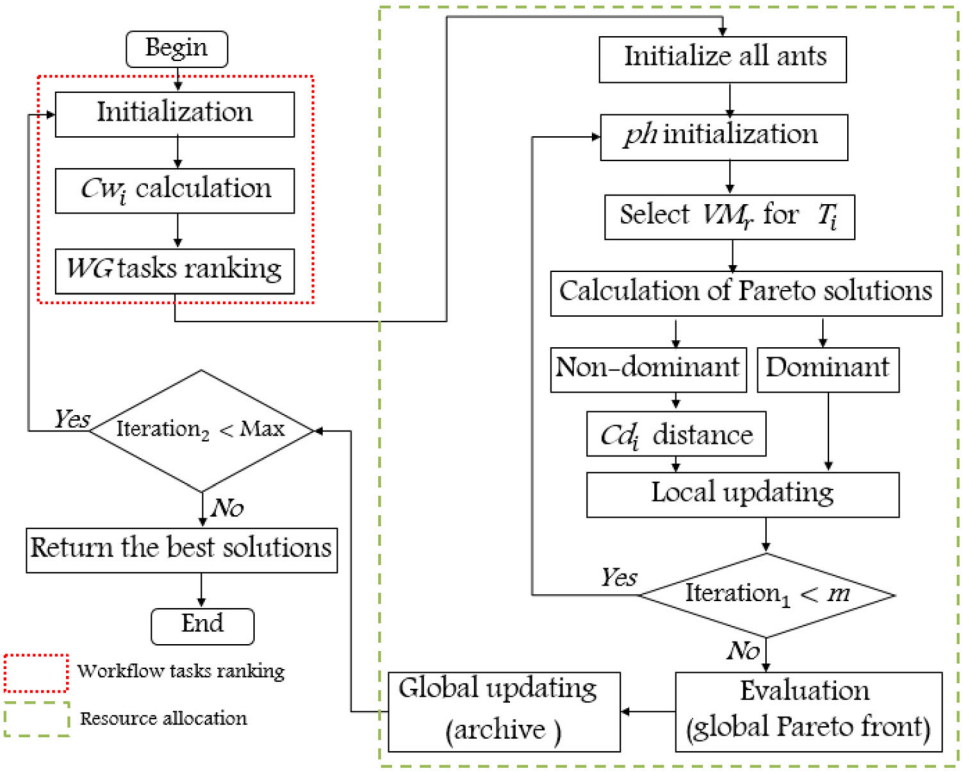


Fig. 2 HEFT-ACO diagram

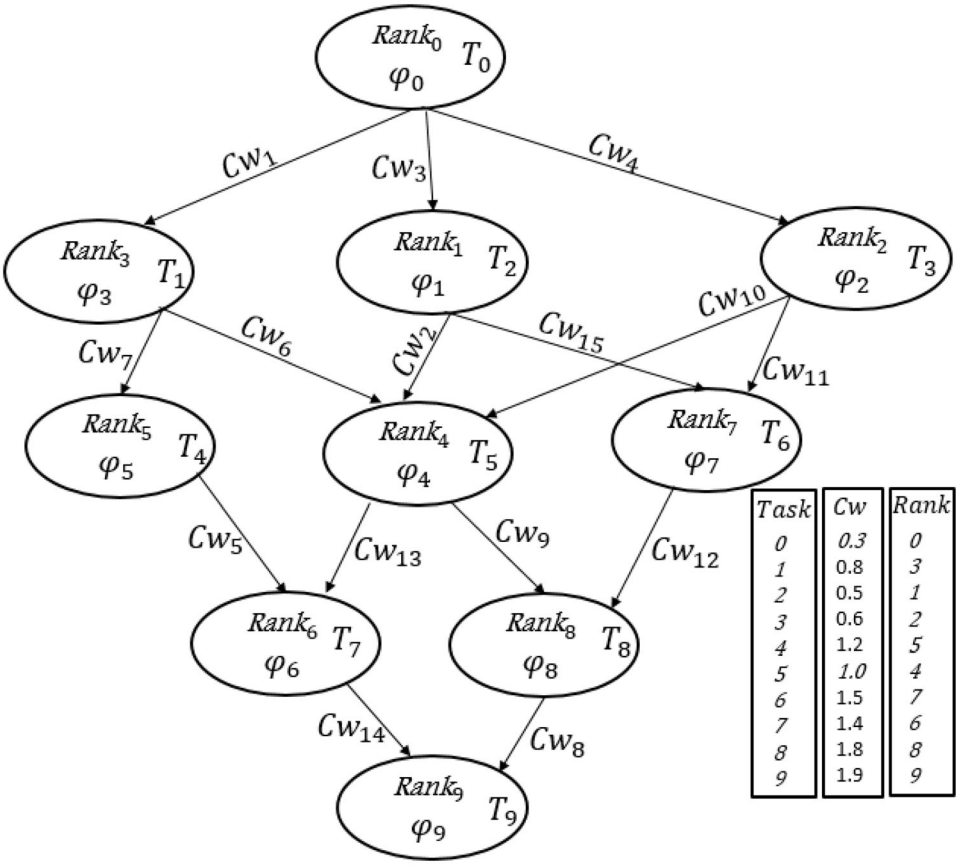


Fig. 3 An example of tasks rank on a workflow application

$$\varphi_{ir} = \begin{pmatrix} \varphi_{1,1} & \varphi_{1,2} & \cdots & \varphi_{1,m} \\ \varphi_{2,1} & \varphi_{2,2} & \cdots & \varphi_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \varphi_{n,1} & \varphi_{n,2} & \cdots & \varphi_{n,m} \end{pmatrix}, \quad \phi_{ir} = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,m} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \phi_{n,1} & \phi_{n,2} & \cdots & \phi_{n,m} \end{pmatrix} \tag{12}$$

4.2 Resources allocation phase

After determining the execution order of each task, the second phase starts by integrating the Ant Colony Optimization (ACO) in order to find a better assignment. This approach was introduced by Dorigo in 1992 [42]. ACO principle is based on the walk of ants amid their colony and the food source. During this movement, the pheromone intensity on the passages increases with the number of ants passing through and drops with the evaporation of pheromone. In this manner, ants find the shortest track to the food.

The main ACO algorithm steps are the construction of the solution, the management of traces of pheromones, and additional techniques such as local search. Additionally, data structures and parameters must be initialized as well as some statistics factors must be kept, as discussed in the following subsections.

4.2.1 Data initialization

In this step, the ants are initialized where each of them represents a class instance comprising the two already mentioned matrices (cost and execution time). We keep the same order of tasks established during the first phase. The pheromone matrix is initialized according to Eq. 16. While the `assignment list` is initially null. The number of

ants representing the size of the `ants' list`. It is determined according to preliminary tests.

By analyzing the behavior of our algorithm during experiments, we found that it can give better performance when the number of ants equals the number of VMs (n). This because if the number of ants is lower than n can lead to a local optimum; since a VM that does not have the best result in each iteration will have a low probability of being selected. While if the number is greater than n, it considerably increases the complexity, slows down the convergence of the algorithm, and does nothing more for performance. This step places all ants at the starting VMs randomly. The initial set of ants used during the optimization is preserved in the following matrix:

$$an = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \tag{13}$$

where: $a_{i,j} \in [0, 1]$.

4.2.2 Solutions construction

Each ant in the initialized list (matrix 13) independently builds its own allocation plan in an asynchronous manner. The solution is established in a constructive way such that the tasks are traveled in order of priority. In other words, the ant examines the probability (P_{ij}) of the current mapping of a selected task on a virtual machine in the set of ready VMs based on the formula 14. After that, the roulette wheel method is used to choose the matching VM for the current task, then keep the selected VM to the taboo table.

$$P_{ij} = \frac{ph_{ij}^{\alpha} * E\varphi_{ij}^{\frac{\beta}{2}} * E\phi_{ij}^{\frac{\beta}{2}}}{\sum ph_{ij}^{\alpha} * E\varphi_{ij}^{\frac{\beta}{2}} * E\phi_{ij}^{\frac{\beta}{2}}} \tag{14}$$

- ph_{ij} the value of the pheromone.
- $E\varphi_{ij}$ the estimated time of the earliest execution times.
- $E\phi_{ij}$ the cost calculated from the estimates of the earliest starts and finish.

To maintain the balance between the heuristic factors (ph, $E\varphi$, $E\phi$), β and α are random natural numbers. In our case, we experimentally use the computation of the earliest and later execution finish time; instead of classical heuristic methods (widely used in the literature [23, 43]). However, we opted for the earliest finish execution time estimation, which gave the best results.

4.2.3 Non-dominated solutions extraction

Since our algorithm deals with a multi-objective problem, the choice of a solution is a decisive step. Seeing that we do not have a priori knowledge of user preferences (minimizing the duration at the expense of the cost or vice versa), we cannot apply the classical methods such as (weighted aggregation, minimization of regret, constraint method, etc). These methods transform the multi-objective problem into one dimension [44].

In this work, the Pareto principle is adopted for the allocation of resources to workflows in a Cloud environment. To describe the concept of Pareto, the following definitions are used:

- `Definition 1 (Pareto dominant)` A decision vector $x^{\in}S$ is said to be Pareto optimal if there is no other decision vector $x \in S$ such that $f_i(x) \leq f_i(x^{)}$, $\forall i \in \{1, 2, \ldots, k\}$ and $f_i(x) < f_i(x^{)}$ for at least one $i \in \{1, 2, \ldots, k\}$ [45].
- `Definition 2 (Pareto front)` The Pareto front (Y^f) is defined as $Y^f = \{f(x) \in R^k, x \in P^o\}$, where P^o is called the Pareto optimal set, and $f(x)$ is the entire feasible solutions in the objective space.

For each iteration, the solutions that belong to the Pareto front are extracted. These solutions will be used to retain a single ant according to the selection criterion chosen at the crowding distance.

- `Crowding distance` (Cd) It is an estimate of the density around a particular solution on the Pareto front, introduced in the NSGA II algorithm [46]. It is calculated according to the following equation:

$$Cd_i = \frac{\varphi(i+1) - \varphi(i+1)}{\varphi_{max} - \varphi_{min}} + \frac{\phi(i+1) - \phi(i+1)}{\phi_{max} - \phi_{min}} \tag{15}$$

The best solution according to this criterion is the one with the greatest distance. This distance is adopted after its experimentally better results given to the detriment of other metrics. The Crowding distance criterion draws its strength from the fact that it favors the exploitation of non-congested regions; this avoids premature convergence (falls on a local optimum).

4.2.4 The global Pareto front construction

Although our algorithm has a convergent aspect, the multi-objective nature makes membership in the list of the global Pareto front possible by solutions of intermediate iterations. For each iteration, the local Pareto front adds to the global list. Once the set of all the iterations is finished, the global Pareto front is calculated. The best ant on the global front will also be selected according to the crowding distance, and its allocation plan will be executed. Knowing that the ants are reinitialized at the end of each iteration.

4.2.5 Pheromone management

In order to attract the ants toward the higher fitness path and to obtain the optimal solution as frequently as possible, it is required to manage the pheromones as follow:

- **Initialization of the pheromone** The pheromone of each ant is initialized by formula 16. This makes it possible to avoid premature convergence and the unjustified excess of the iterations number.

$$ph = \frac{n}{\eta} \tag{16}$$

  where n is the number of ants, and η is the length of a path generated. Indeed, if the initial values of the pheromones are too low, the search is quickly biased by the first rounds generated via the ants. This generally leads to the exploration of the lower areas of the search space. Correspondingly, if the initial values of the pheromone are too high, then many iterations are lost while waiting for the evaporation of the pheromones to further reduce its quantities; so that the pheromone added by the ants can start to guide the search.
- **Pheromone update** After each iteration, the pheromone traces are updated, first reducing the value of the pheromones on all edges with a constant factor (Eq. 17). In this manner, when an ant reaches the determined VM for all tasks, the pheromone on the chosen path of the matching scheme is updated `locally`, using the following rule:

$$ph_{ir}^{i} = ph_{ir}^{i-1} * (1 - \rho) \tag{17}$$

  where ρ indicates the pheromone volatility per unit time. $1 - \rho$ specify the degree of residual pheromone, knowing that $1 < \rho < 0$. The greater the ρ , the faster the pheromone volatilizing. On the other hand, to increment the pheromone time of the path between T_i and VM_j, we add pheromone on the edges of the best locally chosen ant. This leads to the update on the `global` optimal scheme. The update rule is as follows:

$$ph_{ir}^{i} = ph_{ir}^{i-1} + \frac{\lambda}{\varphi_{ir} + \phi_{ir}} \tag{18}$$

  where λ is a constant greater than 0 ($\lambda > 0$).

4.3 Stop criteria

The program stops if the termination condition is reached. We adopt a maximum number of iterations as a stop criterion. This mainly relies on the size of the workflow (proportional to the number of workflow tasks). So, if the current number of iterations is less than the maximum number of iterations, the taboo table will be cleared and return to the data initialization step. Otherwise, the iteration is terminated, and the best solution will be displayed. Algorithm 1 shown the pseudocode of the proposed approach.

Algorithm 1: HEFT-ACO

Input: WG, antList, VM=$\{VM_1, VM_2, ..., VM_n\}$,α , β
Output: Return best solutions $x_i = (T_i, VM_j)$

1: **While** $(Iteration_1 < Max)$

Workflow tasks ranking

2: **For Each**$(i = 0 : m)$
3: $x_i \leftarrow (T_i, VM_j)$
4: Calculate execution times $\varphi_{ir}(x_i)$ using the eqution 1
5: Calculate execution costs $\phi_{ir}(x_i)$ using the equation 6
6: Calculate Cw_i using the formulat 9b
7: **End For**

8: **For Each** $(i = 0 : m)$
9: $RankList \leftarrow Rank(T_i)$ using the equation 9
10: **End For**

Resources allocation

// Initialization of ants (List of ants)
11: **For Each**(ant)
12: $ant_i \leftarrow VM_r$
13: **End For**

14: **While** $(Iteration_2 < m)$
15: Initialization of ph_i using the equation 14
16: Select VM_r for T_i
17: Calculating Pareto solutions (Definition 1)
18: **If** (x_i: non-dominant)
19: Calculate Cd_i using the equation 13
20: best_ants $\longleftarrow$ best solution by crowding distance(Pareto front) (Definition 2)
21: **End If**
22: Locale updating (antList) (Rule 15)
23: **End While**

24: Globale_Pareto_FrontList $\longleftarrow$ add (Pareto front) (Global updating (Rule 16))
25: $T_i \longleftarrow$ define VM machine according to (best_ants)

26: **End While**

5 Implementation and results

5.1 Experiment environment

In the experiments, the WorkflowSim-1.0 toolkit is used. The approaches are coded in Java language and executed on an Intel (R) Core (TM) i7 processor with 3.00 GHz and 6 GB of RAM. The workflowSim is an extension of CloudSim, which offers the possibility to simulate an environment dedicated to the execution of the scientific workflow. It is proposed by "Weiwei Chen" and "Ewa Deelman" from the Institute for Information Studies in Southern California University. WorkflowSim follows the pegasus model which essentially relies on three components: a Workflow Mapper, a Workflow Engine to manage data dependencies, and a Workflow Scheduler to match tasks to resources instantly [47]. The simulations of our approach and the other approaches are made under the same conditions to establish an objective comparison of the results. The experiments are done on three real-world workflow applications from diverse scientific areas. They are Montage workflow for astronomical physics, Cybershake workflow for earthquake hazards, and Ligo workflows for detecting gravitational waves [48].

5.1.1 Environment setup

To evaluate the performance of the HEFT-ACO algorithm, the IaaS provider is modeled by providing a single data center and different types of VM (Table 3). VMs are configured based on Amazon EC2 instances [49]. The storage of each VM is considered large enough to support all of the allocated tasks, while the average bandwidth between virtual machines changes from 5, 10, and 25 Gb/s. Moreover, the virtual machine preparation time is considered between 1 and 1.5 s.

5.1.2 Parameterization of algorithms

The performances of the proposed HEFT-ACO algorithm are compared to the standard ACO, PEFT-ACO, and FR-MOS algorithms. They are implemented according to the works [11, 25, 50] but constructively considering the parameters and formulations of this paper. PEFT-ACO is selected because it gives optimized results in terms of makespan and cost. While the FR-MOS algorithm has shown good behavior for multi-objective problems related to makespan and cost. Appropriate parameter values are determined based on preliminary experience. Therefore, the parameters are determined as α = 1, β= 1, λ=2, and number of iterations = 350. ACOs do not need a large population that the population size set, as the number of ants = 10 is commonly used [11]. The same configuration for all algorithms is kept to ensure an objective comparison. In each experiment, the test is repeated ten times then the average value is taken.

5.1.3 Wilcoxon test

Wilcoxon signed-rank test is used to check if two samples represent two different populations. It is a nonparametric procedure used in hypothesis testing situations and involving a two-sample design. This evaluation represents a pairwise test that includes a null hypothesis H_0 and another hypothesis H_1, where H_0 is a declaration of non-significant difference between the two algorithms and H_1 denotes the presence of a significant difference between the two algorithms [51]. The difference between the performance values of the two algorithms on a problem with n values is denoted d_i. The differences are classified according to their absolute values. Let R^+ be the sum of the ranks for the problems in which the first algorithm has surpassed the second (Eq. 19), and R^- the sum of the ranks for the opposite (Eq. 20). Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored [51].

$$R^+ = \sum_{d_i > 0} rank(d_i) + 1/2 \sum_{d_i=0} rank(d_i) \tag{19}$$

$$R^- = \sum_{d_i < 0} rank(d_i) + 1/2 \sum_{d_i=0} rank(d_i) \tag{20}$$

During the test, a level of significance is set to 0.05. If the p-value (determined from the R^+ and R^- scores) is less or equal than 0.05, then H_0 is rejected in favor of H_1. But, if the value of p is greater than 0.05, then H_0 is not rejected.

Table 3 Types of used virtual machines

Type	CPU	MIPS	Cost ($/h)	BW (Gb/s)
a1.medium	1	4400	0.0255	10
a1.large	2	8800	0.051	10
a1.xlarge	4	17,600	0.102	10
t3.xlarge	4	17,600	0.1664	5
m5n.2xlarge	8	35,200	0.476	25
m5zn.3xlarge	12	52,800	0.991	25

5.2 Optimization metrics

As our algorithm represents a multi-objective approach based on the Pareto principle, we chose Q-metric and FS-metric to analyze the compromise solutions and compare them to the results of the other algorithms (ACO, PEFT-ACO, FR-MOS). In other words, the quality of Pareto-optimal fronts is compared with those of other algorithms.

5.2.1 Q-metric

The metric Q is used to measure the convergence of non-dominated solutions found by two algorithms; It is calculated according to Eq. 21 defined in [52].

$$Q(A,B) = \frac{|\varphi|}{|\gamma|} \tag{21}$$

where A and B represent two sets of Pareto-optimal solutions found respectively by two different algorithms [52]. γ is the set of solutions not dominated in A $\cup$ B and φ = A $\cap$ γ. The Pareto-optimal front found by one algorithm has better convergence towards the true Pareto-optimal front than that found by the other algorithm, if and only if $Q(A,B) > 0.5$ or $Q(A,B) > Q(B,A)$.

5.2.2 FS-metric

The metric FS indicates the size of the space covered by the optimal Pareto front found by an algorithm [53]. We calculate FS-metric as follow:

$$Fs = \sqrt{min_{(x_1,x_2)\in A}(\varphi_i(x_1) - \varphi_i(x_2))^2 + min_{(x_1,x_2)\in A}(\phi_i(x_1) - \phi_i(x_2))^2} \quad (22)$$

A large value of FS-metric is preferable, which means that the optimal Pareto solutions found by an algorithm are widely distributed along the true Pareto front.

5.3 Results and discussions

In the first series of experiments, each objective (cost or makespan) is evaluated independently. The algorithms are executed for three workflow types (Montage, Cybershake, Ligo) with a number of tasks equal to 100 and 300. The results of the average of the 100 executions carried out for each type are mentioned in Table 4.

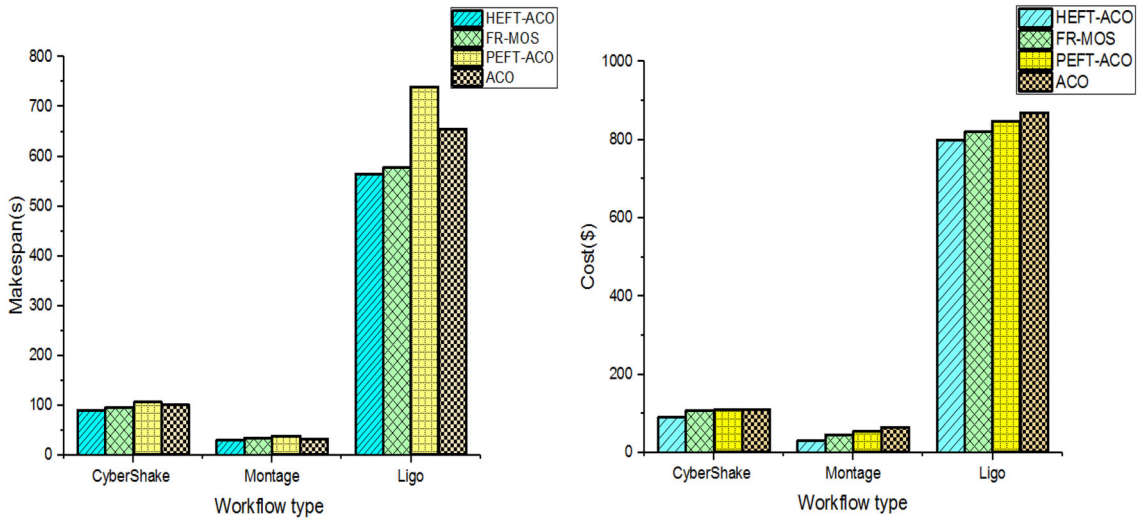
From the results obtained, we can see that the standard ACO algorithm has poor performance in terms of makespan and overall cost for the three types of workflows. In addition, the HEFT-ACO approach outperforms the two other algorithms (PEFT-ACO, FR-MOS) in terms of makespan and cost for all types of workflows. The difference is more significant in terms of makespan. Indeed, HEFT-ACO can search the solution space more efficiently and globally. Also, it relies on "crowding and right optimal front-end" processing. These make it possible to obtain a low cost and a short implementation time.

Figure 4 visualizes the makespan and cost average results for 500 tasks. The graphs confirm that our algorithm surpasses the other algorithms in terms of cost and even makespan. It can be seen that the makespan of HEFT-ACO is the minimum. However, it shows acceptable results in terms of cost.

The second series of experiments is the statistical tests. Fig. 5 shows the trade-off cost over the makespan for three different workflow types of size 400 tasks. From the graphs, HEFT-ACO shows a good trade-off compared to

Table 4 Evaluation of the makespan and cost averages of the algorithms

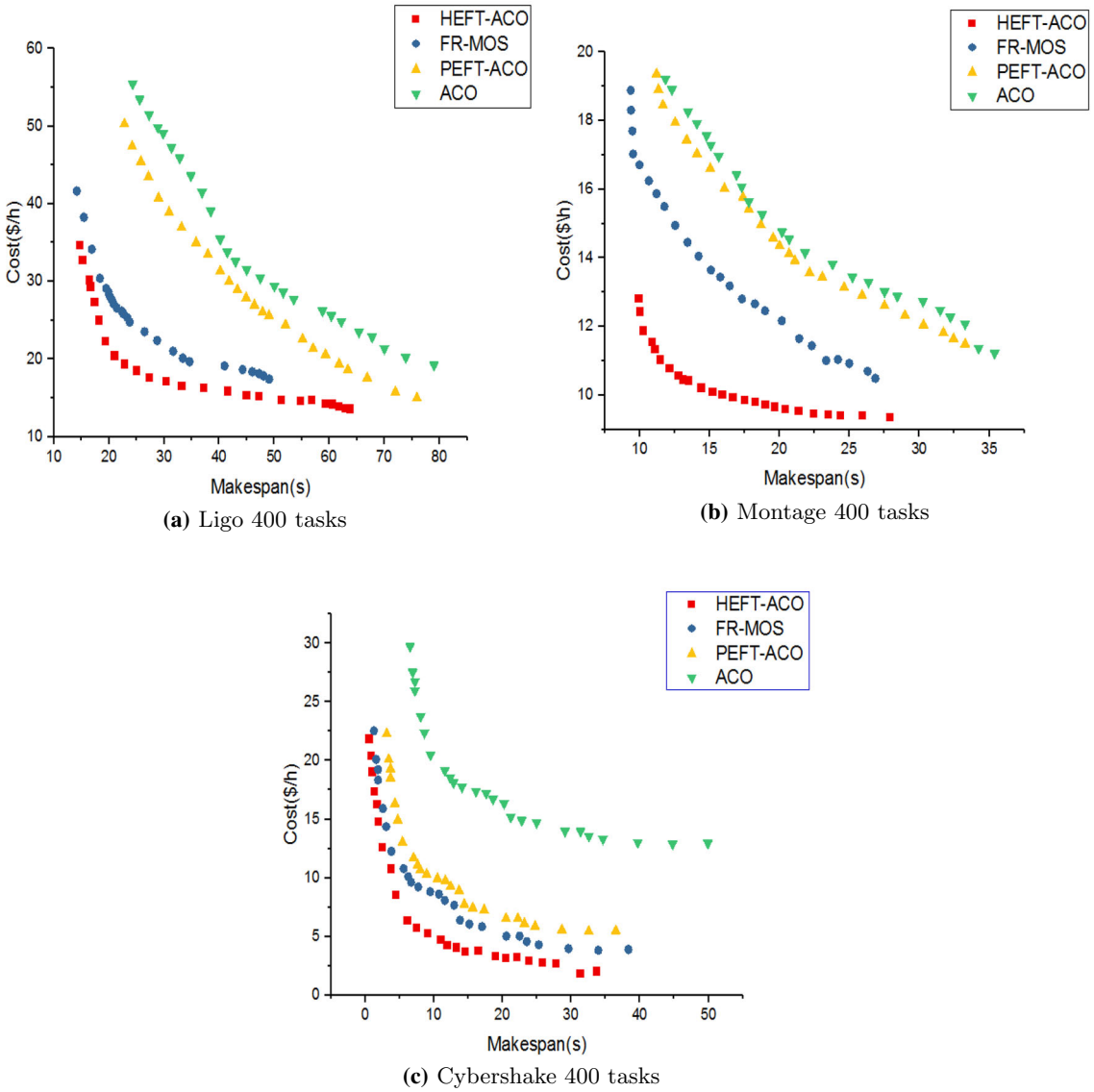
Workflow type	Tasks number	ACO		PEFT-ACO		FR-MOS		HEFT-ACO	
		Makespan (s)	Cost ($/h)	Makespan (s)	Cost ($/h)	Makespan (s)	Cost ($/h)	Makespan (s)	Cost ($/h)
Ligo	100	710.86	78.70	695.87	69.70	609.50	46.94	567.19	33.48
	300	896.73	77.23	888.09	64.41	823.41	42.85	702.53	27.95
Cybershake	100	695.05	70.27	578.56	57.73	513.73	40.70	495.91	22.19
	300	872.42	53.51	701.82	43.96	659.43	32.16	635.93	20.07
Montage	100	654.38	52.20	625.86	25.64	614.78	15.97	473.96	14.99
	300	708.32	30.10	693.49	20.19	679.43	13.84	595.64	9.44



(a) The makespan

(b) The cost

Fig. 4 Makespan and costs average for 500 tasks



(a) Ligo 400 tasks

(b) Montage 400 tasks

(c) Cybershake 400 tasks

Fig. 5 The trade-off comparison of algorithms

the other algorithms. This reflects the ability of the proposed approach to make the best decisions in the existing tasks on the available virtual machines. Generally, it allows giving a good compromise between cost and execution time duration in the allocation process. More precisely, the proposed approach shows a slight improvement compared to FA-MOS for Ligo and Cybershake due to the complex shape of the latter, as well as there is great interdependence between the tasks. However, the improvement in HEFT-ACO remains significant, especially in Montage.

Table 5 shows the test results for three types of workflow with varying the number of tasks where "$>$" meaning that HEFT-ACO surpasses the other algorithm with a significant difference. "$<$" signifies that the second algorithm surpasses our algorithm with a significant difference. "$=$" signifies that there is no sign of active difference between the two algorithms. It presents the R^+, R^-, and p-values computed for all the pairwise comparisons concerning HEFT-ACO (the p-values have been computed by using SPSS).

From this table, we can obviously see that the HEFT-ACO algorithm has a higher number of "$>$" then the other comparative algorithms with the absence of signs "$=$" and "$<$" (only "$<$" with some exceptions). This means that HEFT-ACO is different from the compared approaches according to the Wilcoxon rank sum test. Clearly, it shows significant improvement over ACO, PEFT-ACO, and FR-MOS.

Table 6 shows the value of Q-metric and the value of FS-metric for a number of tasks equal 200 and 900. They are calculated according to Eqs. 21 and 22 , respectively. The value of Q-metrics Q(HEFT-ACO) is "true" for three

Table 5 Wilcoxon test results

Workflow type	Tasks number	HEFT-ACO|ACO			HEFT-ACO|PEFT-ACO			HEFT-ACO|FR-MOS		
		R^+	R^-	p-value/result $*10^{-3}$	R^+	R^-	p-value/result $*10^{-3}$	R^+	R^-	p-value/result $*10^{-3}$
Ligo	50	34	16	0.32/>	41	8,9	0.021/>	28	22	23.014/<
	200	195	95	0.0253/>	177	23	0.0145/>	123	77	0.0488/>
	600	434	166	0.0371/>	417	183	0.05 />	332	268	0.0465 />
	1000	890	110	0.058/>	731	269	0.029/>	519	481	0.048/>
Cybershake	50	42	6,09	0.0492/>	41	9	0.0171/>	33	17	0.138/>
	200	180	20	0.034/>	135	47	0.012/>	147	53	0.0278/>
	600	553	47	0.0147	551	49	0.026/>	438	162	0.045/>
	1000	978	22	0.0141/>	978	125	0.048/>	558	442	0.056/>
Montage	50	43	7	0.0463/>	47	3	0.84/>	23	22	84.148/<
	200	113	87	0.0261/>	156	44	0.0267/>	106	94	0.0148/>
	600	578	22	0.0378/>	488	112	0.0173/>	356	244	0.0434/>
	1000	751	249	0.0324/>	738	262	0.033/>	604	396	0.0236/>

Table 6 Q and FS metrics results

Workflow type	Tasks number	Q/FS metrics	HEFT-ACO	FR-MOS
Ligo	200	Q-metric	/	True
		FS-metric	0.29	0.20
	900	Q-metric	/	True
		FS-metric	0.43	0.34
Cybershake	200	Q-metric	/	True
		FS-metric	0.52	0.47
	900	Q-metric	/	True
		FS-metric	0.7	0.63
Montage	200	Q-metric	/	True
		FS-metric	0.81	0.74
	900	Q-metric	/	True
		FS-metric	0.97	0.87

types of workflows. This indicates that the Pareto solutions of the HEFT-ACO have a ranking each time higher than in the FR-MOS algorithm, which implies that the HEFT-ACO algorithm has the best result in terms of convergence of the multi-objectives. As regards FS-metric, the HEFT-ACO approach has higher values compared to FR-MOS for the considered workflows. The FS values of HEFT-ACO are higher than those of FR-MOS, which led to the conclusion that our approach generates better diversity than FS-MOS.

In most tests, HEFT-ACO significantly outperforms the compared approaches. HEFT-ACO conserve pheromone and heuristic information for every VM, which explains the long time spent when dealing with Ligo workflow. However, HEFT-ACO maintains its efficiency in small and long scale workflows. Also, it outperforms the other three although the variety of tasks number. This may be because HEFT is a greedy heuristic, so it can easily handle the local optimum in large-scale problems, while HEFT-ACO keeps diversity well to examine the search space. The execution time of montage workflow is low because it has many parallel tasks, which results in large availability of VMs.

6 Hypothesis and limitations

The significance of HEFT-ACO is strongly reflected in the multi-objective aspect. It is very effective in dealing with high-quality compromise solutions and providing rapid convergence for resource allocation problems. Another importance appears in the way of classifying workflow tasks. It allows creating an adapted ranking list of tasks, organized in the order of how they should be executed; thus gets rid of a task dependency constraint. From the cloud user perspective, makespan and cost are very important,

which are used as significant metrics for quality of service in cloud computing. The proposed approach showed a high proportion of improvement in workflow scheduling compared to other algorithms. This is an important point for developing cloud-based systems, which are highly growing nowadays. Furthermore, the efficient trade-off between execution time and execution cost could enhance resource utilization, which is a primary metrics that must be ameliorated in recent years.

Correspondingly, the main limitations are that the behavior of HEFT-ACO on load balancing was not taken into account in this study. Its impact is also assessed just for three types of workflows. Moreover, this work does not consider the properties of workflow such as balanced or unbalanced (asymmetric) workflow graph. On the other hand, this study does not consider an operational model for fault-tolerant. In addition, power consumption can be enhanced by adjusting the state of virtual machines. Likewise, the QoS can be improved by introducing the deadline factor.

7 Conclusion

Resource allocation is crucial for Cloud Computing, especially with regard to workflows. In this paper, we have addressed the issue of resource allocation for workflow in cloud computing to improve the quality of services. We have proposed an approach named HEFT-ACO, which is a mixture between two algorithms to minimize makespan and cost at the same time. ACO algorithm is adopted to optimize resource allocation, while HEFT is employed to deal with the dependency of workflow tasks. We have designed the pheromone update rules to deal with non-dominant solutions, thus maintaining diversity and ensuring research efficiency. In addition, this approach used the Pareto technique for multi-objective optimization and crowding distance, which makes it efficient to solve the studied problem.

The experiments are performed using instances of the Amazon EC2 cloud platform and three types of real workflows from different scientific fields. HEFT-ACO is evaluated with standard ACO, PEFT-ACO, and FR-MOS. The results demonstrate that HEFT-ACO is a promising approach to address the QoS issues associated with resource allocation for workflow in the cloud. It outperforms all other algorithms for most carried out tests. This is confirmed by the results obtained by the non-parametric 'Wilcoxon' test; it shows efficient performance compared to other algorithms regarding the trade-off between makespan and cost.

As a perspective, we plan to expand the set of goals to include other metrics, such as fault tolerance. We will increase the number of tasks and VMs in the experiments, testing other types of workflow, and consider further factors such as cloud users' satisfaction (deadline). The deployment of our solution on a real Cloud or multi-cloud environment will also be a future step in our work.

References

1. Buyya, R., Broberg, J., Goscinski, A.M.: Cloud Computing: Principles and Paradigms, vol. 87. Wiley, Hoboken (2010)
2. Arregoces, M., Portolani, M.: Data Center Fundamentals. Cisco Press, Indianapolis (2003)
3. Smith, J.E., Nair, R.: The architecture of virtual machines. Computer **38**(5), 32–38 (2005)
4. Belgacem, A., Beghdad-Bey, K., Nacer, H., Bouznad, S.: Efficient dynamic resource allocation method for cloud computing environment. Clust. Comput. **23**, 1–19 (2020)
5. Belgacem, A., Beghdad-Bey, K., Nacer, H.: Dynamic resource allocation method based on symbiotic organism search algorithm in cloud computing. IEEE Trans Cloud. Comput. (2020). https://doi.org/10.1109/TCC.2020.3002205
6. Noor, T.H., Zeadally, S., Alfazi, A., Sheng, Q.Z.: Mobile cloud computing: challenges and future research directions. J. Netw. Comput. Appl. **115**, 70–85 (2018)
7. Belgacem, A., Beghdad-Bey, K., Nacer, H.: Task scheduling in cloud computing environment: a comprehensive analysis. In: Proceedings of the International Conference on Computer Science and its Applications, pp. 14–26. Springer (2018)
8. Sprinks, J., Wardlaw, J., Houghton, R., Bamford, S., Morley, J.: Task workflow design and its impact on performance and volunteers' subjective preference in virtual citizen science. Int. J. Hum.-Comput. Stud. **104**, 50–63 (2017)
9. Momenzadeh, Z., Safi-Esfahani, F.: Workflow scheduling applying adaptable and dynamic fragmentation (WSADF) based on runtime conditions in cloud computing. Future Gen. Comput. Syst. **90**, 327–346 (2019)
10. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
11. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
12. Gawali, M.B., Shinde, S.K.: Task scheduling and resource allocation in cloud computing using a heuristic approach. J. Cloud Comput. **7**(1), 4 (2018)
13. Jiang, H., Song, M., et al.: Dynamic scheduling of workflow for makespan and robustness improvement in the iaas cloud. IEICE Trans. Inf. Syst. **100**(4), 813–821 (2017)
14. Arabnejad, V., Bubendorfer, K., Ng, B.: Budget and deadline aware e-science workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. **30**(1), 29–44 (2018)
15. Na, W., Zuo, D., Zhang, Z.: Dynamic fault-tolerant workflow scheduling with hybrid spatial-temporal re-execution in clouds. Information **10**(5), 169 (2019)
16. Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., Shiyan, H.: Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft. Future Gen. Comput. Syst. **93**, 278–289 (2019)
17. Rehman, A., Hussain, S.S., Zia ur Rehman, S.Z., Shamshirband, S.: Multi-objective approach of energy efficient workflow scheduling in cloud environments. Concurr. Comput. Pract. Exp. **31**(8), e4949 (2019)

18. Haidri, R.A., Katti, C.P., Saxena, P.C.: Cost-effective deadline-aware stochastic scheduling strategy for workflow applications on virtual machines in cloud computing. Concurr. Comput. Pract. Exp. **31**(7), e5006 (2019)
19. Gupta, S., Agarwal, I., Singh, R.S.: Workflow scheduling using Jaya algorithm in cloud. Concurr. Comput. Pract. Exp. **31**(17), e5251 (2019)
20. Xue, S., Peng, Y., Xiaolong, X., Zhang, J., Shen, C., Ruan, F.: Dsm: a dynamic scheduling method for concurrent workflows in cloud environment. Clust. Comput. **22**(1), 693–706 (2019)
21. Zhang, H., Zheng, X., Xia, Y., Li, M.: Workflow scheduling in the cloud with weighted upward-rank priority scheme using random walk and uniform spare budget splitting. IEEE Access **7**, 60359–60375 (2019)
22. Gao, Y., Zhang, S., Zhou, J.: A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud. IEEE Access **7**, 125783–125795 (2019)
23. Sinha, N., Srivastav, V., Ahmad, W.: Deadline constrained workflow scheduling optimization by initial seeding with ant colony optimization. Int. J. Comput. Appl. **155**(14), 24–29 (2016)
24. Jethava, A.N., Desai, M.R.: Optimizing multi objective based dynamic workflow using aco and black hole algorithm in cloud computing. In: Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 1144–1147. IEEE (2019)
25. Farid, M., Latip, R., Hussin, M., Hamid, N.A.W.A.: Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. IEEE Access **8**, 24309–24322 (2020)
26. Han, P., Chenglie, D., Chen, J., Xiaoyan, D.: Minimizing monetary costs for deadline constrained workflows in cloud environments. IEEE Access **8**, 25060–25074 (2020)
27. Adhikari, M., Amgoth, T., Srirama, S.N.: Multi-objective scheduling strategy for scientific workflows in cloud environment: a firefly-based approach. Appl. Soft Comput. **93**, 106411 (2020)
28. Al-Janabi, S., Mohammad, M., Al-Sultan, A.: A new method for prediction of air pollution based on intelligent computation. Soft Comput. **24**(1), 661–680 (2020)
29. Al-Janabi, S., Alwan, E.: Soft mathematical system to solve black box problem through development the farb based on hyperbolic and polynomial functions. In: Proceedings of the 2017 10th International conference on developments in eSystems engineering (DeSE), pp. 37–42. IEEE (2017)
30. Ali, S.H.: Novel approach for generating the key of stream cipher system using random forest data mining algorithm. In: Proceedings of the 2013 sixth international conference on developments in esystems engineering, pp. 259–269. IEEE (2013)
31. Al-Janabi, S., Salman, A.H.: Sensitive integration of multilevel optimization model in human activity recognition for smartphone and smartwatch applications. Big Data Mining Anal. **4**(2), 124–138 (2021)
32. Al-Janabi, S., Alkaim, A.F.: A nifty collaborative analysis to predicting a novel tool (drflls) for missing values estimation. Soft Comput. **24**(1), 555–569 (2020)
33. Al-Janabi, S., Alkaim, A.F., Adel, Z.: An innovative synthesis of deep learning techniques (dcapsnet & dcom) for generation electrical renewable energy from wind energy. Soft Comput. **24**(14), 10943–10962 (2020)
34. Alkaim, A.F., Al-Janabi, S.: Multi objectives optimization to gas flaring reduction from oil production. In: Proceedings of the International conference on big data and networks technologies, pp. 117–139. Springer (2019)
35. Alkaim, A.F., Al-Janabi, S.: A comparative analysis of dna protein synthesis for solving optimization problems: a novel nature-inspired algorithm. Adv. Intell. Syst. Comput. **1372** (2020)
36. Kliazovich, D., Pecero, J.E., Tchernykh, A., Bouvry, P., Khan, S.U., Zomaya, A.Y.: Ca-dag: modeling communication-aware applications for scheduling in cloud computing. J. Grid Comput, **14**(1), 23–39 (2016)
37. Lee, Y.C., Han, H., Zomaya, A.Y., Yousif, M.: Resource-efficient workflow scheduling in clouds. Knowl. Based Syst. **80**, 153–162 (2015)
38. Malawski, M., Figiela, K., Bubak, M., Deelman, E., Nabrzyski, J.: Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Sci. Program. **2015**, 5 (2015)
39. Maciej, M.: Cost-and deadline-constrained provisioning for scientific work flow ensembles in iaas clouds. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press (2012)
40. On line: Amazon ec2 instance store. Accessed (2 Jun 2021). (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/)
41. Zhao, H., Sakellariou, R.: An experimental investigation into the rank function of the het- erogeneous earliest finish time scheduling algorithm. In: Proceedings of the European Conference on Parallel Processing, pp. 189–194. Springer (2003)
42. Dorigo, M., Di Caro, G.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)
43. Zhou, Y., Huang, X.: Scheduling work flow in cloud computing based on ant colony opti- mization algorithm. In: Proceedings of the 2013 Sixth International Conference On Business Intelligence And Financial Engineering, pp. 57–61. IEEE (2013)
44. Tabucanon, M.T.: Multiple Criteria Decision Making in Industry, vol. 8. Elsevier Science Ltd, New York (1988)
45. Giagkiozis, I., Fleming, P.J.: Pareto front estimation for decision making. Evol. Comput. **22**(4), 651–678 (2014)
46. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput **6**(2), 182–197 (2002)
47. Chen, W., Deelman, E.: Workflowsim: a toolkit for simulating scientific work flows in distributed environments. In: Proceedings of the 2012 IEEE 8th International Conference on E-Science, pp. 1–8. IEEE (2012)
48. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Future Gen. Comput. Syst. **29**(3), 682–692 (2013)
49. On line: Amazon ec2 on-demand pricing. Accessed (24 Apr 2021). (https://aws.amazon.com/ec2/pricing/on-demand/)
50. Kaur, A., Kaur, B..: Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment. J. King Saud Univ. Comput. Inf. Sci. (2019)
51. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)
52. Wei, J., Zhang, M.: A memetic particle swarm optimization for constrained multi-objective optimization problems. In Proceedings of the 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 1636–1643. IEEE (2011)
53. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)

Ali Belgacem currently serves as an Associate Professor in the Department of Computer Science at the University of M'hamed Bougara in Algeria. He received his Ph.D. degree from Military Polytechnic School, Algeria, in 2020. His project of research investigates to determine an optimal solution for dynamic resource allocation in cloud computing environments.

Kadda Beghdad-Bey Bey received his M.S. degree in Industrial Computer Science from Military Polytechnic School in 2003. He received his Ph.D. in Computer Science from USTHB University, Algiers in 2010. Current, He is the Professor and Head of the Laboratory of Distributed and Complex Systems at M. P. School. His areas of research includes biometrics identification, parallel and distributed computing, Meta-heuristics techniques and task scheduling, and resources allocation in cloud commuting.