Multimedia Tools and Applications (2023) 82:33015–33038
https://doi.org/10.1007/s11042-023-14739-w

PPTS-PSO: a new hybrid scheduling algorithm for scientific workflow in cloud environment

Adnane Talha [1] · Mohammed Ouçamah Cherkaoui Malki [1]

Received: 18 March 2022 / Revised: 13 June 2022 / Accepted: 4 February 2023 /
Published online: 4 March 2023

Abstract
The use of complex scientific workflows in cloud computing environments, taking into account different interdependency criteria, is becoming a key objective for cloud service providers and for customers. This gives the task scheduling operation a higher priority in order to improve the quality of services. In this work, we introduce a novel hybrid PPTS-PSO algorithm based on two efficient algorithms with the goal of improving the scheduling phase of a set of interdependent tasks that make up scientific workflows in the cloud-computing platform with the best execution time and cost while staying within the deadline and budget constraints. An intelligent variant of the PSO algorithm named neighborhood PSO and the heuristic PPTS algorithm are used. The suggested method can assign tasks in scientific workflows to the most appropriate cloud virtual machine. Therefore, our strategy takes into account resource allocation too. The experimental results show that our solution overcomes different algorithms in the literature with minimum iterations.

Keywords Scheduling · Cloud computing · PSO · Scientific workflows · Meta-heuristic algorithm

1 Introduction

Cloud computing (CC) has grown as a study topic in recent years, giving a cost-effective deployment framework for hosting and running workflows. It is considered the leading model for distributed computing due to its elasticity, speed, and pay-per-use model. The benefits of CC include scalability, adaptability, and cost effectiveness. Aside from these benefits, it also has certain disadvantages. One of the most serious concerns is security, as data that is stored

✉ Adnane Talha
adnane.talha@usmba.ac.ma

Mohammed Ouçamah Cherkaoui Malki
oucamah.cherkaoui@usmba.ac.ma

[1] FSDM, LPAIS Lab, Sidi Mohamed Ben Abdellah University, Fez, Morocco

can be accessed by anybody. Other drawbacks include a lack of standards, technical challenges, and attack vulnerability. The number of cloud service providers (Google, Microsoft Azure, Amazon, and others) continues to grow, resulting in an increase in the number of cloud services. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) are some of the most popular cloud computing services. Infrastructure as a Service (IaaS) is a well-known cloud service platform that gives consumers access to capable and adaptable computational resources. Cloud infrastructure is discharged as virtual computers (VMs). Clients having access to an endless amount of resources for application execution at a lower cost of use. Users can use services for specific needs at a tariff set by their CSP (Cloud Service Provider). These services are available at any time. There are massive benefits about using cloud computing to execute workflows. For starters, it relieves the user of the responsibility of maintaining their own infrastructure. Second, in terms of price, it is reasonable. Third, it allows access from anywhere. To run scientific applications in the cloud environment within a certain number of criteria remains a big challenge for researchers. There are massive benefits about using CC to execute workflows. For starters, it relieves the user of the responsibility of maintaining their own infrastructure. Second, in terms of price, it is reasonable. Third, it allows access from anywhere. In the scientific field, there are workflows that are sensitive to enormous volumes of information, others that are vulnerable to sophisticated calculations, and still others that are sensitive to multiple criteria concurrently. Therefore, the deployment and hosting of these workflow applications require a robust infrastructure with high-performance computing, communication, and storage. Most previous works designed for grid, cloud or cluster environments consider the fixed amount of resources and focus only on minimizing the execution time. The above forces scientists to develop WF scheduling optimization algorithms that strike the right balance between two main qualities of the Service (QoS) parameters: time and cost. To optimize scientific applications, workflow scheduling (WFS) is the ideal technique for distributing workflow tasks to computer resources. WFS aims to manage the execution of interdependent tasks by considering precedence constraints on resources. This problem is known as NP-complete. This strategy prompted the researchers to provide a near-optimal solution. The heuristic approach and the meta-heuristic approach are the two types of scheduling strategies for workflow graphs [33] (Topcuoglu et al., 2002). Heuristic algorithms are built on simple, fast, and easy-to-implement rules, but they frequently result in local solutions because they rely on a number of constraints provided by domain experts. They are better suited to straightforward optimization tasks. In order to efficiently deal with a bigger search space while designing large-scale applications, meta-heuristic techniques are presented. Each meta-heuristic method comes with its own set of upsides and downsides. Meta-heuristics is a random search technique that aims to find a near-optimal solution within a reasonable time. This implies a relatively longer computation time. Multiple meta-heuristics methodologies have been proposed, including the gravitational search algorithm (GSA) [11, 28], the ant colony optimization (ACO) algorithm [13], the particle swarm optimization (PSO) [20], the artificial bee colony (ABC) [19], the dragonfly algorithm (DA) [27] and the genetic algorithm (GA) [16, 31]. We observed that in the scientific simulation disciplines, there are workflows that are sensitive to huge amounts of data, others that are sensitive to complex operations, and still others that are sensitive to multiple criteria at once. Therefore, we have thought about the way to give a solution aimed at optimizing the processing of these SWfs and specifically, finding a middle ground between two diametrically opposed quality of service (QoS) parameters: time and cost. In general, qualitative metrics such as computational time and cost are used to determine QoS. This inconsistency prompted us to propose a realistic

solution targeted at reducing processing time while also lowering processing expenses as much as possible while staying within deadlines and budgetary constraints. The suggested solution's fundamental idea is to match tasks to appropriate resources, in order to reduce computational cost and execution time while keeping the deadline and budget in mind. To achieve this dual target, we combed the literature for the most widely utilized methodologies that achieve this goal while also producing good outcomes. We selected to combine the capability and simplicity of PPTS [12] with the evolutionary algorithm PSO [17]. This mixture allows us to obtain a hybrid solution aiming at the optimization of the scheduling operation. PSO's accuracy can be improved by adding the PPTS generated solution into the initial population of randomly generated solutions. PPTS was chosen due to its excellent SWf scheduling capabilities. This method tends to speed up the scheduling process in order to produce the best results in terms of computation time and cost. The main contributions of this work include the following 2 aspects:

(1) The neighborhood particle swarms strategy is adopted which includes the neighborhood-learning factor, to overcome the constraints of the simple particle network in order to increase the opportunities for exploring more potential solutions.
(2) We develop a new hybrid method for cloud tasks scheduling, that integrate PPTS with PSO together to combine the capability and simplicity of PPTS with the evolutionary algorithm PSO.

The Workflowsim tool, an extension of cloudSim is used to evaluate the performance of our algorithm for some common workloads in simulated data centers. We show that this approach exceeds existing strategies for attempting to tackle the WFS problem in cloud environments depending on the results of our simulation studies. The rest of this paper is organized as follow. In Section 2, we briefly review a few similar works. In Section 3 we formulates the problem of scheduling the scientific workflow in CC with the target of decreasing cost and time while remaining in time and under budget. While in Section 4, we present a hybrid strategy based on PPTS and PSO. Sections 5 and 6 present the outcomes of the experiments and the conclusion.

2 Related work

Over the years, many studies have tried to offer solutions to the workflow-scheduling problem, and as previously said, a range of heuristic and meta-heuristic methodologies have been researched and evaluated. Many enhancements to these algorithms were developed to deal with limited scheduling conditions. Common heuristic algorithms include Heterogeneous Earliest Finish Time (HEFT) [33], Critical Path on Processor (CPOP) [33], PEFT [5] and PPTS [12].

The HEFT [33] and CPOP [33] approaches are two major scheduling methods that try to deliver the best performances while saving scheduling time for a certain number of heterogeneous processors. Using an insertion-based mechanism, the HEFT technique sends the task with the smallest ascending rank value to the processor with the shortest completion time at each level. The second algorithm, known as CPOP, prioritizes tasks by adding the bottom-up and top-down rank values. Another distinction is the processor selection phase, which assigns critical tasks to the processor with the shortest total execution time. In this paper [5], the authors utilized a list-based scheduling technique known Predict Earliest Finish Time

methodologies (PEFT). The process includes a look-ahead function into an optimistic cost table despite minimizing the computation's temporal complexity (OCT). The result is an optimistic cost because processor availability is not taken into consideration in the calculation. The approach is entirely dependent on the OCT table, which is used to prioritize workloads and choose processors. The work puplished in [12], introduce the Predict Priority Task Scheduling (PPTS) methodologies which is a list-based scheduling mechanism, by including a prediction function into both phases of the PPTS algorithm, the major goal is to shorten the scheduling length. In [17], the primary idea behind the proposed algorithm GHEFT is to combine the advantages of genetic and HEFT algorithms while reducing their downsides. The program assigns priorities to each subtask using the HEFT algorithm, and then searches for a task-processor mapping solution using a genetic approach.

The research in [18], proposes a reformed scheduling strategy based on a pre-allocated energy consumption level for unassigned jobs, as well as an energy consumption limitation mechanism. The purpose of the study published in [2] is to develop a list scheduling with task duplication (LSTD) technique for the amount of time needed to accomplish workflow applications. The LSTD incorporates a task duplication methodology to the list scheduling technique despite having low total amount of time complexity. In [30], the authors suggest an upgrade to HEFT in which the heuristic makes locally optimum judgments based on estimations of a single job, looks ahead in scheduling, and considers information about the influence of the decision on the children of the allotted work. In addition, the authors in [25] Using only a state-space clustering method, this work has proposed the optimal task scheduling with task duplication. It also gives important new definitions of typical boundary parameters in the perspective of duplication. In the paper [1], by using the modified antlion optimizer algorithm, a new multi-objective optimization solution for work scheduling difficulties in cloud computing systems with balanced job configuration/distribution was developed (MALO). Because it mixes the characteristics of genetic algorithm methodologies (GA) and an evolution strategy, the DE algorithm was also used in collaboration with a local search strategy and a differential evolution (DE) mechanism to optimize the ALO's exploitation search-ability (ES). In [8] the authors suggested a workflow scheduling method focused on particle swarm methodology. Competitive aspects such as makespan, load balance, resource utilization, and speedup ratio are reviewed by the fitness function proposed. The particle is modelled so that a complete solution can be generated while maintaining dependence limitations. In [6], the authors proposes a hybrid Min-Min (MM) and RoundRobin (RR) strategy named (HMMRR) to improve resource utilization and system performance by lowering average response time and system latency while reducing the makespan (execution time) of all virtual machines. In [7], The researchers in this paper announced HPSOGWO, a new hybrid multi-objective method that combines the functions of two well-known methodologies, particle swarm optimization algorithm (PSOA) and grey wolf optimization (GWO). The authors of paper [21] discuss their algorithm named PSO + LOA, a combined optimization model for scheduling processes in the cloud that merges particle swarm with the lion's eye optimization approach (LOA) to improve running time while preserving within budget constraints. Using spectral division and subdivision of the input task graph, [32] describes a two-phase hybrid task scheduling technique. G-SP distributes every portion of the directed graph to a low-power processor to reduce power consumption. In [29] The suggested approach is a mixture of the recently discovered SMO algorithm and the other widely used heuristic algorithm BDSD, which is a budget and deadline-constrained algorithm that aids HSMO in developing a workable program. Additionally, the suggested technique uses a penalty function to limit the number of solutions that do not meet the QoS restrictions. In [35]

Offers an alternative list-based scheduling technique to schedule tasks described as a DAG form. The main purpose of this technique is to schedule jobs to the appropriate processing node in a fog environment because fog nodes' computation capacity is limited. The computing cost and even the node's completion time must be properly considered when distributing tasks to the fog node. In paper [23] merges the traditional particle swarm optimization methodology with a significantly increased ant-lion optimization (ALO) algorithm. During the scheduling phase, the cloud data is secured using the Data Encryption Standard security mechanism (DES). The study in [4] propose two hybrid metaheuristic algorithms titled DE-SA and GA-SA that are also matched with a ravenous approach based on the genetic algorithm (GA), differential evolution (DE), and simulated annealing (SA). The research in [3] summarizes existing surveys on scientific workflow management systems and cloud computing scheduling. It includes a taxonomy of scientific workflow applications as well as their properties. It demonstrates how established scientific workflow management and scheduling strategies, such as resource scheduling, fault-tolerant scheduling, and energy efficient scheduling, function. It goes over numerous performance evaluation factors and platforms that are used to evaluate scientific workflow management. It identifies evaluation platforms for evaluating scientific workflow management approaches based on a variety of performance evaluation factors and presents several technical requirements for presenting new scientific workflow management strategies. The authors of the study in [22] address the problem that existing cloud schedulers consider only a single resource (RAM) when co-locating workloads, resulting in SLA violations owing to non-optimal VM placement. The nova scheduler has been changed to provide a multi-resource based VM placement technique to increase application performance with respect of CPU utilization and execution time. The work in [34] investigate a computational paradigm in which each machine has a bounded capacity to perform a set number of tasks at the same time. Based on the above-mentioned paradigm, the Extended High to Low Load (ExH2LL) task scheduling heuristic is presented, which aims to balance workload among accessible computing resources while enhancing resource usage and minimizing the makespan. ExH2LL determines task-to-machine assignment automatically based on the current load on all devices.

3 Problem formulation

A. *Workfow model*

In this paper, we will deal with scheduling a set of interdependent tasks (workflow) in a cloud-computing infrastructure to minimize the total execution time (makespan) and the cost while respecting the constraints of limited execution time and budget. The representation of the workflow can only be done by using a suitable technique. For this purpose, we choose a directed graph that has no circuits and whose arcs are directed. A workflow application W = (T, E) is modelled by a directed acyclic graph (DAG) (Fig. 1), where T is a set of tasks T = $\{t_1, t_2 \ldots t_m\}$, m = |T| in the workflow. E is a sequence of directed edges $\{e_{i,j} | (t_i, t_j) \in E\}$ that describes the interdependence of data and control between t_i and t_j. The task t_i in which is said to be a parent (predecessor) task of t_j and t_j is said to be a child (successor) task of t_i. The task t_j can proceed its execution only if all its ancestors tasks have finished their execution and the associated data or parameters have been transferred from these tasks to t_j. The parent task of an edge is the source task, and the child task is the target task. Succ(t_j) and pred(t_j) represent the successor of t_j and the predecessor of t_j respectively. A task without a parent task is called an

input task, while a task without a child is called an output task. Figure 1 shows an illustration of a typical workflow DAG scheme.

B. *The scheduling problem*

In this part, we will present the different axes of the task-scheduling problem. To do this, we will start with the system, application, and scheduling models. In our study, we focused on several main components described in Fig. 2. This model includes a broker, datacenters, hosts, VMs and tasks.

The cloud service provider has a series of data centers modeled according to the operating systems, CPU architecture, hypervisor, available network bandwidth, usage costs, and virtual machine allocation policies. In addition, our cloud data center comprises heterogeneous physical virtual machines. Each VM has a CPU, which can be multi-core and whose performance is defined in millions of operations per second (MIPS). In addition, user requests will be handled in multiple VMs whose resource requirements are measured in MIPS, amount of RAM, and communication bandwidth. The broker operates as a mediator between the users and the data centers and provides the appropriate level of quality of service. It enables for the selection of data centers where the VMs will be deployed and defines how the tasks will be managed and in which VM will be executed. Each tasks in our model is modeled using several characteristics such as the size in MIPS, the length of the input and output files exchanged between the provider and the VMs, the bandwidth, and the size occupied by the task in memory. The image size, number of CPUs, processing capacity in MIPS, quantity of RAM,bandwidth, hypervisor types and task scheduling mechanism are all part of the VM's basic configuration. In our paradigm, the execution time for every task is a variable depending on the configuration of the VM, and the execution can be estimated through the profiling system provided by the data center. Suppose that in distributed systems all processors are identical. In that situation, each processor's execution time for a given task is the same. However, in real distributed systems, processor speed will vary. The heterogeneity model depicts the difference in processing speeds required to complete a particular task. The processing speed in distributed systems is unpredictable in reality, and the formula below would be used to calculate the degree of imbalance: $h \in [0...1)$.

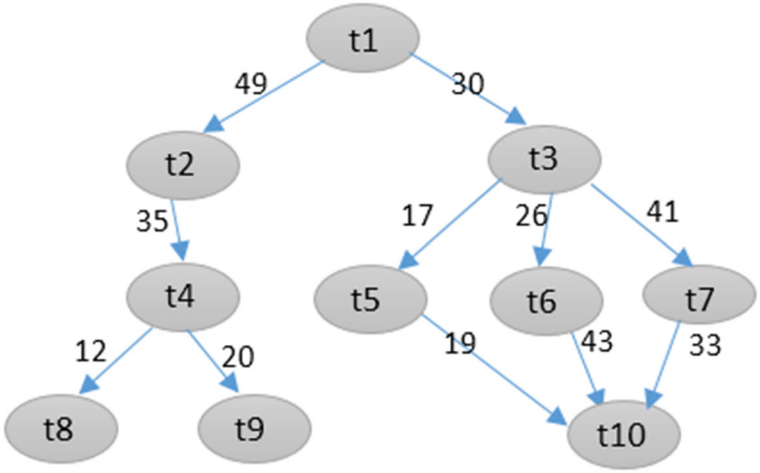

Fig. 1 Sample workflow DAG scheme Task scheduling modeling AND formulation

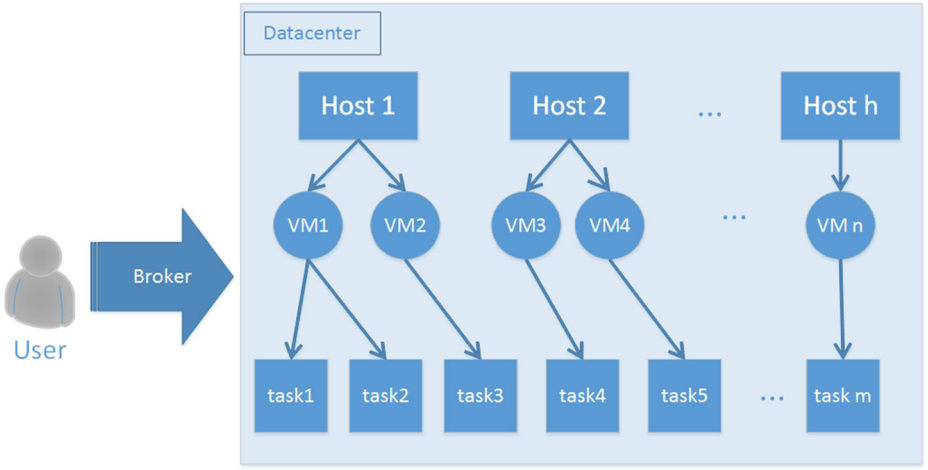

Fig. 2 System architecture for PPTSPSO

$$\textbf{degree of imbalance} = \frac{h+1}{1-h} \tag{1}$$

Where h is the heterogeneity parameter. If we take $h = 0$, the degree of imbalance is 1, or if we take $h = 0.5$, the degree of imbalance is 3. This indicates that the fastest processor in the distributed system can complete a task three times faster than the slowest processor. Each task mapped to a particular VM has an estimated computational cost based on a time interval, which is the unit of measurement for the cost calculation, we assume in our paper that resources are charged per unit time of use. The term "quantum" refers to this time interval. The fastest virtual machine should by definition, be the most expensive. Table 1 shows all symbols used in our paper.

C. *Particle swarm optimisation*

PSO is ranked among the best optimization algorithms suggested by the two scientists Kennedy and Eberhart [12] in 1995. Each individual (particle) represents a solution to the situation and is characterized by a position and a velocity. The algorithm modifies these two values as it goes along, bringing them closer to the desired results. Each particle is declared in the form of a vector X (x_1, x_2, …, x_d) representing its position in the space. We record the velocity, position, and the value of the fitness function it has achieved, which refer to the position in each iteration. At each particle generation, our algorithm produces the most advantageous global position named gbest and the best personal position called pbest. Their fitness function values return the selection of these two elements. During the search method, the following two equations are employed to measure the particles velocity and position:

$$\mathbf{V_i(t+1)=w.V_i(t)+C_1.r_1*(pbest-x_i(t))+C_2.r_2*(gbest-x_i(t))} \tag{2}$$

$$\mathbf{x_i(t+1)=x_i(t)+V_i(t)} \tag{3}$$

Equation (2) represents the velocity of the ith particle at iteration t. Where w is the inertia parameter. $\mathbf{C_1}$ and $\mathbf{C_1}$ are constants acceleration factors, $\mathbf{r_1}$ and $\mathbf{r_2}$ are randomly initialized between 0 and 1. $\mathbf{x_i}$ represents the current position of particle i. pbest is the ideal position of the particles reached in the past, gbest is the ideal position of the particle swarm. Equation (3) returns the position x of the i-th particle for the t-th iteration. Figure 3 presents the flowchart of the PSO algorithm.

Table 1 Shows the different symbols used in our article

Table 2: Symbols definition

N	The number of tasks
P	The number of VMs.
M	The number of particles.
$\{t_1, t_2, \ldots, t_N\}$	The set of N arrival tasks.
$\{v_1, v_2, \ldots, v_P\}$	The set of p VMs.
$mips_j$	The processing rate of the VM v_j
$ET_{i,k}$	The execution time that v_k needs to process t_i
$size_i$	is the size of the task t_i
Quantum	Discrete unit to calculate the cost of using VM
CC_k	The computing capacity of v_k (MFLOPS)
TTij	The data transfer time between $t_i$and t_j
BW	Average bandwidth
FT_k	Is the unit time to release the v_k (lease end time)
ST_k	The lease start time as long as the execution of a workflow
TCw	The total cost of executing all the tasks.
LPT_k	The rental price per unit of time for v_k
γ	Is the smallest unit for calculating the cost of using the VM.
D	is the time limit to execute all the tasks
B	Is the estimated budget.
pbest	Personnel best position of a particle.
gbest	Global best position of a particle.
best	The best position of the particle's neighborhood
C_1, C_2, C_3	Acceleration constants.
r_1,r_2, r_3	Randomozid numbers in [0.1[.
w	Inertia weight
$x_i(t)$	Particle's position
$V_i(t)$	Particle's velocity
Mw	The total time spent to complete the total tasks (Makespan).
EFT(ti, vj)	earliest completion time of ti on vj

D. *Optimisation by neighborhood particle swarms.*

The basic particle swarm optimization technique relies on individual cognition and social behavior to select the next position, which leads to the problem of all particles migrating to the optimal overall position. As a result, we will apply the updated particle swarm optimization [36], which includes the neighborhood-learning factor, to overcome the constraints of the simple particle network in this paper. Particles adjust their velocity and position in the neighborhood particle swarm optimization method depending on individual behavior, group behavior, and the optimum individual experience in the neighborhood. Following the previous argument, we change the particle swarm's velocity update formula to:

$$\mathbf{V_i(t+1)=w.V_i(t)+C_1.r_1*(pbest-x_i(t))+C_2.r_2*(gbest-x_i(t))+C_3.r_3*(Nbest-x_i(t))} \quad (4)$$

$$\mathbf{x_i(t+1)=x_i(t)+V_i(t)} \quad (5)$$

With C_1, C_2 Are acceleration factors in the standard optimisation algorithm, while C_3 is a neighborhood acceleration factor. r_1, r_2 *and* r_3 are random values in the range [0.1].

The current position of particle i is represented by $\mathbf{x_i}$. pbest is the best particle position achieved in the past. Gbest is the best particle swarm position and Nbest is the best particle neighborhood position. The best position, called Nbest(best closest) is chosen based on two

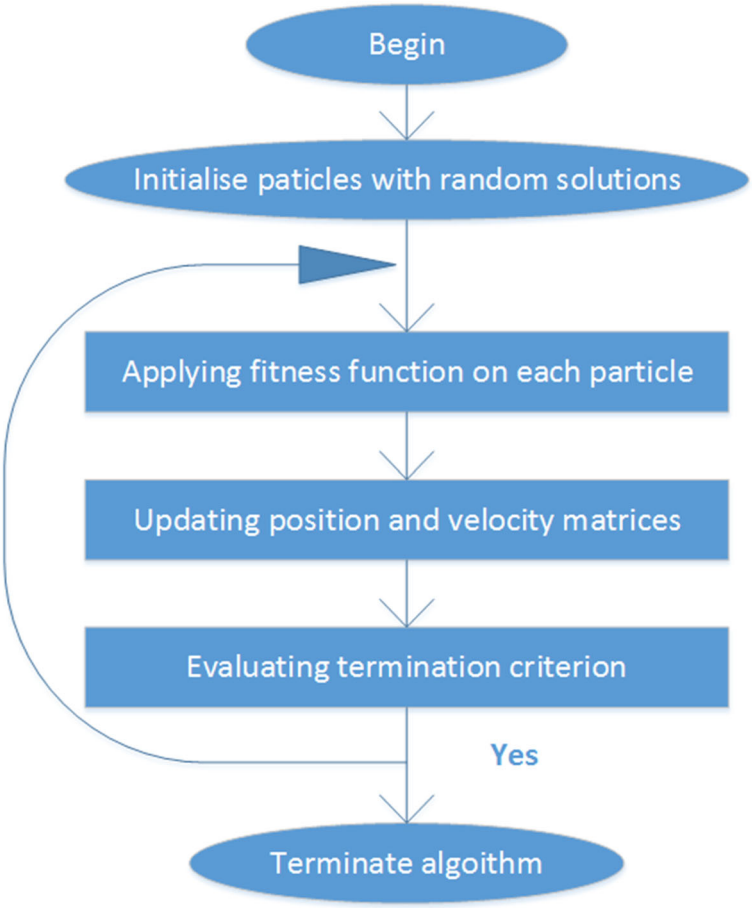

Fig. 3 Flowchart of PSO algorithm

factors: first, it must be adjacent to the current particle, and second, it must have visited a position with higher fitness. To do this, we propose to choose as the best neighbor particle of the current particle i, for each iteration d, the one, which, among all the particles of the swarm, except the particle i, maximizes the following ratio (Fitness-Distance):

$$\frac{\boldsymbol{Fitness}(\boldsymbol{P}_j)-\boldsymbol{Fitness}(\boldsymbol{X}_i)}{|\boldsymbol{P}_{jd}-\boldsymbol{X}_{id}|} \tag{6}$$

Where $\boldsymbol{P_j}$ and $\boldsymbol{X_i}$ are respectively the right position of particle j, and the best position of the current particle i.

E. *PPTS algorithm*

PPTS is a new heuristic algorithm for workflow scheduling introduced by djigal et al. [12]. The capacity of PPTS to reduce the complexity and length of the scheduler by looking ahead is a key advantage. It takes into account the present task's execution time as well as the execution

times of all its immediate predecessors in its formula, all without adding to the complexity of the algorithm when compared to other similar algorithms. The algorithm is based on the Predictive Cost Matrix (PCM), which is used to calculate the priorities of each task and the processor selection phase using the look-ahead concept. We will start by defining the main elements of this algorithm, starting with the PCM matrix, which is a (t × p) matrix, where each PCM(t_i, p_j) represents the most prominent element of the shortest paths between task t_i and the exit task. The element of PCM is determined by recursively solving the following formula:

$$\mathbf{PCM}(\mathbf{t_i}, \mathbf{v_j}) = \mathbf{max}_{\mathbf{t_k} \in \mathbf{succ}}(\mathbf{t_i}) \left[\mathbf{min}_{\mathbf{v_\gamma} \in \mathbf{P}} \left\{ \mathbf{PCM}(\mathbf{t_k}, \mathbf{v_\gamma}) + \mathbf{ET}(\mathbf{t_k}, \mathbf{v_\gamma}) + \mathbf{ET}\left(\mathbf{t_i}, \mathbf{v_\gamma}\right) + \mathbf{TT}_{\mathbf{i,k}} \right\} \right] \tag{7}$$

Where $\mathbf{TT}_{i,k} = \mathbf{0}$, if $\boldsymbol{v_j} = \boldsymbol{v_\gamma}$ for the exit task PCM($\boldsymbol{t_{exit}}$, $\boldsymbol{v_j}$) = $\mathbf{ET}(\mathbf{t_{exit}}, \boldsymbol{v_j})$. To set the priority of each task t_i, one must first calculate the average of the PCM($\boldsymbol{t_i}$, $\boldsymbol{v_j}$) denoted by $\overline{\boldsymbol{PCM}}(\boldsymbol{t_i})$ defined by (9):

$$\overline{\boldsymbol{PCM}}(\boldsymbol{t_i}) = \frac{\sum_{i=1}^{p} \boldsymbol{PCM}(\boldsymbol{t_i}, \boldsymbol{v_j})}{\boldsymbol{p}} \tag{8}$$

After obtaining the priority list, the tasks are ranked in descending order.

F. *The allocation of virtual machines (VM):*

To allocate tasks to a VM, we need to calculate the value look − Ahead_{EFT} for each task $\mathbf{t_i}$ on a machine $\boldsymbol{v_j}$, which is the sum of EFT(t_i, v_j) and PCM($\mathbf{t_i}$, $\mathbf{v_j}$) defined by:

$$\mathbf{Look\text{−}Ahead}_{\boldsymbol{EFT}}(\boldsymbol{t_i}, \boldsymbol{v_j}) = \boldsymbol{EFT}(\boldsymbol{t_i}, \boldsymbol{v_j}) + \text{PCM}(\boldsymbol{t_i}, \boldsymbol{v_j}) \tag{9}$$

Such that EFT(t_i, v_j) represents the earliest time to end of t_i on v_j. We select the VM that has the lowest value of Look − Ahead_{EFT} to ensure that the successors of the running task finish earlier.

Algorithm 1 PPTS algorithm

1- Compute the PCM matrix as defined in (7) for each task
2- Calculate $\overline{\boldsymbol{PCM}}(\boldsymbol{t_i})$ for each task t_i as given in (8) and arrange them in descending order.
Repeat Step 3-4, until the list is empty.
for all the v_j do
3-Calculate $\boldsymbol{EFT}(\boldsymbol{t_i}, \boldsymbol{v_j})$ value-using insertion based scheduling policy.
4-Using (9) to compute $\mathbf{Look - Ahead}_{\boldsymbol{EFT}}(\boldsymbol{t_i}, \boldsymbol{v_j})$
End for
5 - Allocate task to the VM with minimum $\mathbf{Look - Ahead}_{\boldsymbol{EFT}}$.

4 The proposed algorithme

Researchers face a huge issue when it comes to task allocation in cloud systems. In this topic, several studies and algorithms have been conducted, particularly heuristic algorithms, which can be classed as efficient solutions to this type of problem (the workflow scheduling). In our research, we tried to implement a hybrid algorithm in which they used two sophisticated types of algorithms. The first heuristic algorithm PPTS is used to produce the population of particles passed to the second meta-heuristic algorithm named PSO. Our study used an optimization algorithm that manages the initialization phase of the particle population intelligently, unlike other classical algorithms that randomly generate the population. In this stage, we use a heuristic algorithm that elaborates a list of tasks. The result of this algorithm is itself an optimal planning. The major advantage of this technique is that we will get the best optimized planning with resource allocation. Secondly, the number of repeats has been reduced, which reduces the execution time of the algorithm. We are looking for a solution in which the best time/cost trade-off has been applied while respecting certain constraints. Algorithm 1 presents the different steps to build an initial population to implement our PSO-based solution.

A. *Fitness function*

In the conduct of our research, we have addressed two main objectives. The first is to minimize the computational cost, and the second aim is to minimize the execution time of the workflow. To achieve these goals, we need to clearly define a fitness function that meets this. Our fitness function returns solutions of such a kind that the best solution means it had a better score while the worst score represents the worst solutions. The algorithm must satisfy the budget (B) and deadline (D) constraints. The function is made of two parts, the first one concerns the makespan execution time and the second one examines the total cost. The function is then defined using the following formula:

$$\begin{cases} \textbf{minimize } (\mathbf{Mw}, \mathbf{TCw}) \\ \textbf{subject to} \end{cases} \mathbf{Mw} < \mathbf{D}, \mathbf{TCw} < \mathbf{B} \tag{10}$$

Where D is the deadline and B is the estimated budget. The particle with the lowest cost and the lowest execution time will be chosen.

1. ***Makespan***

Before define the makepan, we must give some definition. The execution time of the task t_i in $\boldsymbol{v_k}$ is defined as follows:

$$\mathbf{ET}_{i,k} = \frac{\boldsymbol{size_i}}{\boldsymbol{CC_k}} \tag{11}$$

Where $\boldsymbol{CC_k}$ represents the computational capacity of $\boldsymbol{v_k}$ (FLOPS) and $\boldsymbol{size_i}$ represents the task's size. The communication between two tasks t_i and t_j requires a transfer time from the parent t_i to the child t_j. The transfer time noted TT_{ij} is calculated using the following formula:

$$\frac{\boldsymbol{TT_{ij}} = \boldsymbol{Data}\ (t_\mathbf{i}, t_\mathbf{j})}{\mathbf{BW}} \tag{12}$$

Where Data(t_i, t_j) is the size of data that needs to be received from t_i, t_j and BW is network bandwidth. It should be mentioned that the TT_{ij} is equal to zero if t_i and t_j belong to the same VM. Therefore, the internal data transfer will be zero, as in most cloud data centers. For t_i, the longest input transferring time from all its input files can be denoted as:

$$TT_i = max_{t_{j \in pred(t_i)}} (TT_{ij}) \tag{13}$$

Where t_j is one of the predecessor tasks of t_i.

The processing time (PT) of t_i scheduled on v_k is defined as:

$$\mathbf{PT}\ (\ t_i, v_k) = TT_i + \mathbf{ET}_{i,k} \tag{14}$$

The Start Time (ST) of task t_i *on* v_k is calculated as:

$$\mathbf{ST}\ (\ t_i, v_k) = \begin{cases} 0, if\ \ t_i = t_{entry} \\ max_{t_j \in pred(\ t_i)} \{\mathbf{PT}\ (\ t_i, v_k) + \mathbf{ST}\ (\ t_j, v_k)\ \}, otherwise \end{cases} \tag{15}$$

The Finish Time (FT) of t_i *on* v_k can be computed as:

$$\mathbf{FT}\ (\ t_i, v_k) = \begin{cases} \mathbf{PT}\ (\ t_i, v_k), if\ \ t_i = t_{entry} \\ max_{t_j \in pred(\ t_i)} \{\mathbf{FT}\ (\ t_j, v_k) + \mathbf{PT}\ (\ t_j, v_k)\ \}, \quad otherwise \end{cases} \tag{16}$$

The makespan expresses the entire amount of time it take to execute the workflow, which is defined as follows:

$$\mathbf{Mw} = \mathbf{Max}\{\ \mathbf{FT}\ (\ t_i, v_k)\ \}\ \ \mathbf{i} = \mathbf{1}, .., \mathbf{n}\ \ \mathbf{k} = \mathbf{1}, .., \mathbf{m} \tag{17}$$

2. ***The total cost***

For CSP services, pricing rules are defined, with a predetermined price for unit data transmission between two VMs and a pay-per-use charge for processing time units. The cost of execution can be defined as follows:

$$\mathbf{TCw} = \frac{((FT_k) - (ST_k))}{\gamma} * \mathbf{LPT_k} \tag{18}$$

Where $\mathbf{LPT_k}$ is the rental price per unit time for v_k, FCT_k *and* SCT_k are respectively the unit time to release the v_k (end-of-lease time) and the start-of-lease time during the execution of a workflow. The unit γ is the shortest unit for calculating the cost of using the VM.

The cost is determined by taking into account the costs of data centre, processing capacity, memory, bandwidth, storage of each VM instance type. The overall cost is calculated by adding the costs of each task processing.

B. *Hybrid PSO Algorithm*

1. *Encoding*

As seen in Fig. 4, a scheduling solution is represented by each particle in the population. The location of a particle, is an array reflecting the correlation between tasks and virtual machines. This array has a dimension equal to the total number of tasks in our workflow. Each element indicates the index of a VM to execute the corresponding task. For example, Position [13] = 3 means that task t4 will be performed on VM number 3. However, the minimum value that can be had in an array is one, while the highest value is p (the number of VMs). The second array contains the velocity of a particle of the same dimension as the position array. Each element contains the result of an equation for changing the velocity at each iteration. Then, the result is used to change the index of the virtual machines that exist in the first list.

2. *Initialisation*

In this phase, we will show the different steps followed to generate an initial population. Algorithm 1 describes the instructions used, which are based on the PPTS heuristic algorithm.

Algorithm 2 Generate an initial population

Input: Empty population with size *M*
Output: Population of size M particles
1: Generate randomly M particles
2: Integrate the PPTS algorithm into the population
3: generate the population of particles

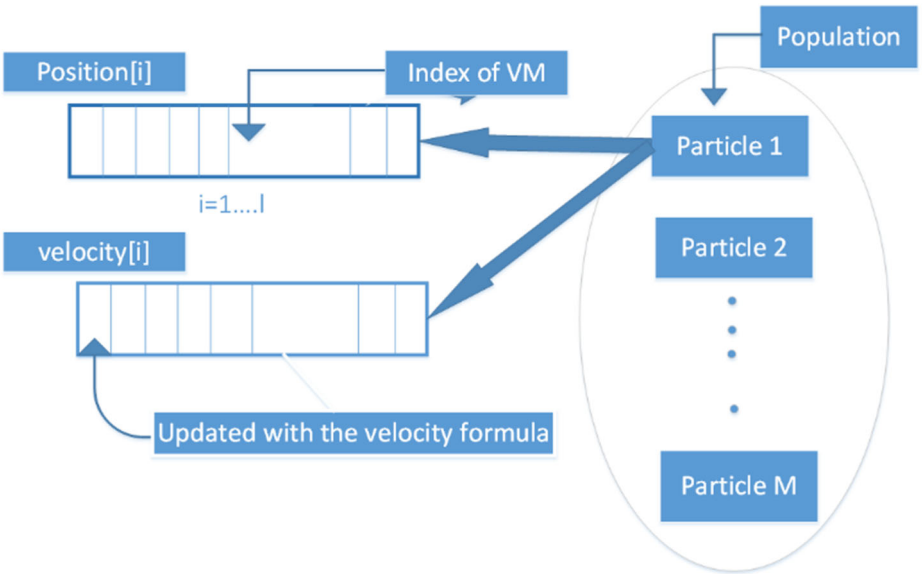

Fig. 4 The coding of particles in our population

3. *The proposed algorithm*

The different instructions of our hybrid algorithm will be explained in this section. Because performance and results are depending on the input, initialization is a crucial phase in any algorithm. The PPTS-PSO algorithm takes a population of particle swarms initialized using the PPTS program as an argument to take advantage of the strengths of both the PPTS and PSO heuristic and meta-heuristic approaches. The following section has a complete explanation of each step:

Algorithm 3 The proposed PPTS-PSO Algorithm

Input: Tasks T , Virtual Machines V
Output: The optimized schedule
Begin
1. T[M] // Initialise the first set of swarm using schedule obtained from PPTS algorithm
2. **For** (i=1;i<=M;i++)
3. {
4. x(t) =T[i]// Initialise the position of each particle with the schedule obtained from algorithm 2
5. V(t) ← random() // Initialise the velocity randomly
6. Pbest ← x_i(t) // Initialise Pbest with the initial position of the particle
7. Gbest ← better(Pbest) // Initialise Gbest with the best of all Pbest among all particle
8. Nbest ← Null // initialize the Nbest
9. }
10. **Repeat**
11. {
12. **For**(i=1;i<=M; i++) // for each particle in the swarm
13. {
14. F ← Fitness_value() //calculate the fitness value for each particle position using equation (10)
15. **If** F(Pi) >F(Pbest) **then** // if the actual value is bigger than Pbest then
16. Pbest ← x_i(t) // update the value of the particle with the current value.
17. End If
18. . **If** F(Pi) >F(Gbest) **then** // if the present value is higher than Gbest
19. Gbest ← x_i(t)// change the particle's value with the current value
20. **End If**
21. **Nbest** is chosen by maximizing the formula (6)
22. V(t+1) = V(t) // Update particle velocity using Equation (5).
22. x(t+1) =x(t) // Update particle position using Equation (4).
23. }// end for
24. **Until** (result converge) // end repeat
25. End

The fitness value will be compared to Pbest and then Gbest at each iteration. If the new values produce superior results, Pbest and Gbest will be adjusted to reflect the new values. The Nbest is chosen by maximizing the formula (6). The schedule generated by algorithm 2 and the number of tasks are inputs to our algorithm. The number of tasks in the process determines the population's size. The tasks in the scientific workflow will be represented by the particles, and the position will be the virtual machine determined. The for loop from lines 2 to 9 will do the needed initialization. At line 4, the particles position is initialized with the position from the algorithm 2. The virtual machine assignment determined by the algorithm 2 is used to initialize the particles. At line 5, the velocity is randomly initialized. The particle's Pbest is initialized with the initial particle's position at line 6. The Gbest is initialized at line 7. The Gbest is established up in a manner that the better position of the whole particles is chosen. The Nbest is initialized too using the null value. Lines 10–26 will be executed for each iterations of the

swarm search. Line 15 calculates the fitness value for each particle. In line 16–18, the fitness value is compared with the Pbest. If a better result is obtained in the search, then the Pbest is updated. Lines 19–21, likewise, look for an enhancement of the Gbest. If a new improvement is discovered, the Gbest will be updated to reflect it. In lines 22, the Nbest is determined through maximization of the formula (6), the velocity and the positions are updated. Using the knowledge gathered in the current iteration, the particles will go in the correct direction. The until loop will be repeated until the Mw and TCw objective values converge.

The time complexity of algorithm 2 is O(p.n^2). Here n is the number of tasks in the workflow W and p is the number of VMs. The first for loop at line 2 iterates only for the number of particles M, which is always equal to n^2. On the other hand, The do–until loop starting at line 10 will be executed for 'q' times. The q value set in the experiment is 25. On an average case, the results converged within 17 iterations for smaller workflows, 19 for medium workflows and around 20 for larger workflows. The for loop starting at line 12 will be iterated for M times. Hence, the time complexity of our algorithm is O((q + p).n^2). Fig. 5 explains all the main elements of the hierarchical architecture used in this paper, which is based on the PSO algorithm. Users must submit their service requests to the cloud provider.

The latter must determine the most effective way of task scheduling. This solution outlines the job sequence and the virtual machines (VM) that execute them at the best possible time and cost. The experiments and simulations used to construct our technique for tackling task-scheduling challenges in the cloud environment are presented in the following part.

5 Performance evaluation

In this section, we present the experimental settings including materials and evaluation metrics used to evaluate the performance and the effectiveness of our algorithm.

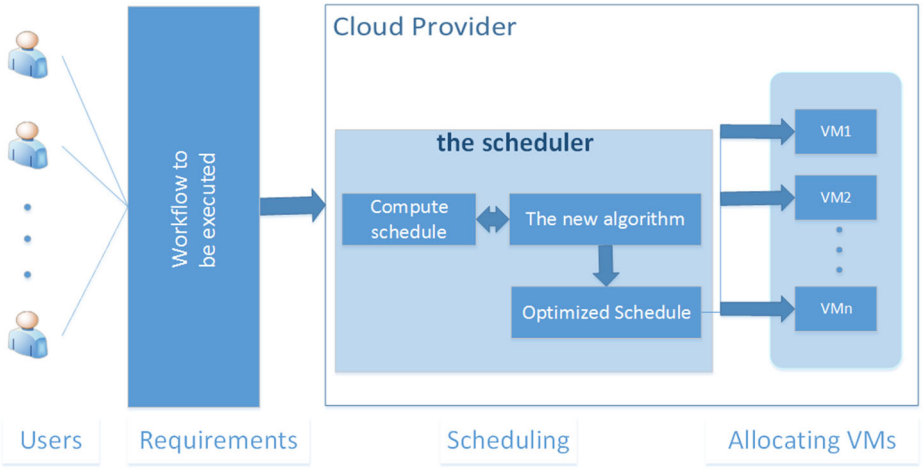

Fig. 5 The architecture of the PPTS-PSO algorithm

Table 2 Configuration setup

Datacenter parameters		VMs parameters		Cost parameters		PSO parameters	
Number of VMs	5	RAM (MB)	512	Processing usage cost	3.0	number of particles	25
Number of cloud users	1	MIPS	Random{500,1000,2000, 3000,4000,5000}	Memory usage cost	0.05	inertia weight	0.9
		BW	1000	Storage usage cost	0.1	C1,C2,C3	2
		Pes Number	1	BW usage cost	0.1		

A. *Simulation setup*

Synthetic workflow data was prepared using the Pegasus workflow repository [26]. Table 2 lists the simulation parameters that we used to test the proposed scheduling algorithm's performance.

B. *Framework simulation environment:*

We used the WorkflowSim Framework [9], which is built on CloudSim, to analyze our suggested solution. An open-source simulator provides a level of workflow management. A workflow planner create a list of tasks that are first given as an XML file in its raw form. The workflow parser module prepares this task list. If necessary, the clustering engine can combine tasks into a set of jobs. The workflow engine must then order these tasks based on the dependence criteria. Before processors run ordered tasks, the workflow scheduler gets involved to match them to available VMs. Montage, Cybershake, Inspiral, and Ligo are four scientific applications families that model real-life data flows. Figure 6 these applications were chosen because of their versatility in terms of application areas and resource needs. Several of these workflows are given to the science establishment through the Pegasus Workflow Mangers platform tools [26], which become scalable and flexible (Pegasus Workflow Management platform 2015). The DAX is an XML-formatted description of an abstract workflow that serves as the model's principal input. It includes a list of all referenced files and all task dependencies, as well as a specification of all jobs. Let α be the unit of time. If the user uses the leased VM partially. It will be considered as a full time use. For example, if $\alpha = 100$ min,

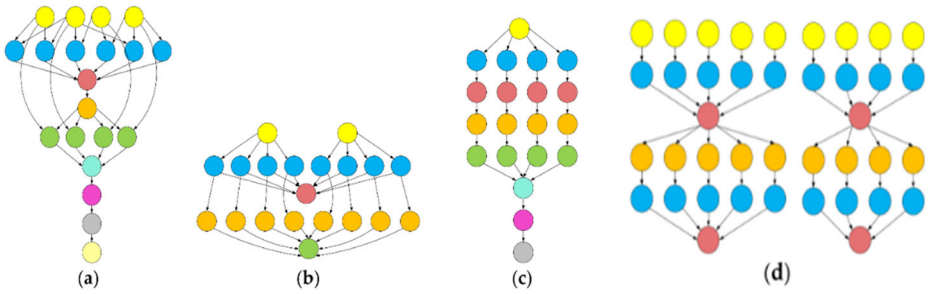

Fig. 6 The structure of workflows: (**a**) Montage; (**b**) CyberShake; (**c**) Epigenomics; (**d**) LIGO and Inspiral Analysis

and the VM is used 101 min then the user will pay 2 periods. i.e. 200 min. The inertia weight value ∝ was adjusted between 0.1 and 1, and it was found that the best results could be obtained when the inertia is set at 0.7–0.9. Our approach converged in 18 iterations for small size tasks (50), 21 iterations for medium size tasks (100), and 24 iterations for big size tasks (1000). The requirement for additional iterations comes from the need to eliminate local minima. Table 3 shows a comparison between these applications in terms of system intensity.

6 Implementation and results

We conducted a set of experiments with existing heuristic algorithms like:

- FCFS [15]
- MinMin [10]
- MaxMin [14]
- RoundRobin [24]
- HEFT [33]
- PPTS [12]
- PSO [20]

The suggested algorithm is developed in Java and runs on the Eclipse platform. The tests have been carried on a workstation with a window OS and 2.5 GHz Intel Core i5 processor and 4GB of RAM. The proposed PPTS-PSO approach is based on a population of 25 particles, which is sufficient to achieve reasonable convergence rate. Tables 4, 5, 6 et 7 show the results of a

Table 3 Comparison between the scientific applications

Scientific WF application	Domain	System intensiveness
Montage	Astronomy	Data-intensive
CyberShake	Earthquake science	Data/memory-intensive
Inspiral	gravitational physics	CPU-intensive
Ligo	gravitational physics	CPU-intensive

Table 4 Experimental results for assembly data sets of Montage 50(small), 100(medium) and 1000(large) tasks

Scheduling algorithm	Makespan			Cost		
	Montage_50	Montage_100	Montage_1000	Montage_50	Montage_100	Montage_1000
FCFS	129,77	256,79	2555,62	1627,44	3427,53	36,003,19
MinMin	132,59	269,97	2555,54	1627,16	3427,81	36,007,12
MaxMin	129,51	256,74	2556,49	1627,96	3429,09	36,009,32
RoundRobin	129,51	256,65	2555,54	1628	3427,99	36,007,12
HEFT	129,28	256,69	2555,99	1627,65	3428,07	36,002,25
PPTS	130,05	257,01	2555,87	1628,11	3428,12	36,005,33
PSO	129,9	256,88	2556,03	1627,66	3429,22	36,008,45
PPTS-PSO	129,2	256,98	2554,25	1627,12	3427,37	36,001,01

Table 5 Experimental results for assembly data sets of cybershake 50(small), 100(medium) and 1000(large) tasks

Scheduling algorithm	Makespan			Cost		
	cybershake _50	cybershake _100	cybershake _1000	cybershake _50	cybershake _100	cybershake _1000
FCFS	372,21	728,99	4618,74	38,316,28	76,698,49	127,841,06
MinMin	389,96	859,15	4634,59	38,315,49	76,697,57	127,837,23
MaxMin	363,07	713,39	4616,18	38,318,36	76,694,85	127,844,3
RoundRobin	378,02	848,78	4642,08	38,315,01	76,701	127,837,85
HEFT	365	726,95	4622,94	38,316,36	76,693,12	127,840,64
PPTS	367	799,56	4625,03	38,317,36	76,691,54	127,841,65
PSO	365,2	745,74	4622,79	38,315,25	76,673,09	127,836
PPTS-PSO	364,84	724,53	4613,77	38,297,91	76,664,48	127,811,51

Table 6 Experimental results for assembly data sets of Inspiral 50(small), 100(medium) and 1000(large) tasks

Scheduling algorithm	Makespan			Cost		
	ligo_50	ligo_100	ligo _1000	ligo _50	ligo_100	ligo _1000
FCFS	2715,77	4444,6	45,610,89	35,468,16	63,427,22	685,498,43
MinMin	3095,91	4598,11	45,795,51	35,468,5	63,427,13	685,495,84
MaxMin	2833,29	4403,09	45,588,97	35,467,97	63,426,57	685,493,23
RoundRobin	2838,68	4470,82	45,610,39	35,468,17	63,426,84	685,496,95
HEFT	2695,84	4390,54	45,464,99	35,467,79	63,426,08	685,496,61
PPTS	2689,26	4530,65	45,420,43	35,467,85	63,426,23	685,497,65
PSO	2698,02	4420,87	45,546,69	35,468,06	63,427,12	685,496,75
PPTS-PSO	2801,64	4370,07	45,103,43	35,467,89	63,425,83	685,491,22

series of studies using Montage workflow, Cybershake workflow, Ligo workflow, and inspiral process to compute the makespan and cost, accordingly.

A. *Performance analysis of real workflows based on makespan and cost*

The experiments presented in Figs. 7, 8, 9, 10, 11 and 12 show that the suggested PPTS-PSO method outperforms current algorithms like PSO, PPTS, HEFT, FCFS, MAXMIN, MINMIN, and ROUNDROBIN in terms of makespan and cost. We considered applications graphs with

Table 7 Experimental results for assembly data sets of Ligo 50(small), 100(medium) and 1000(large) tasks

Scheduling algorithm	Makespan			Cost		
	Inspiral_50	Inspiral_100	Inspiral_1000	Inspiral_50	Inspiral_100	Inspiral_1000
FCFS	2715,77	4444,6	45,693,99	35,468,16	63,427,22	686,720,76
MinMin	3095,91	4598,11	45,774,57	35,468,5	63,427,13	686,719,56
MaxMin	2833,29	4403,09	45,674,34	35,467,97	63,426,57	686,717,56
RoundRobin	2838,68	4470,82	45,873,33	35,468,17	63,426,84	686,719,98
HEFT	2695,84	4390,54	45,716,48	35,467,79	63,426,08	686,719,86
PPTS	2811,12	4396,55	45,711,02	35,469	63,428,86	686,718,36
PSO	2800,46	4399,15	45,588,29	35,468,2	63,427,2	686,720,01
PPTS-PSO	2816,96	4395,98	45,577,07	35,467,06	63,426,94	686,715,33

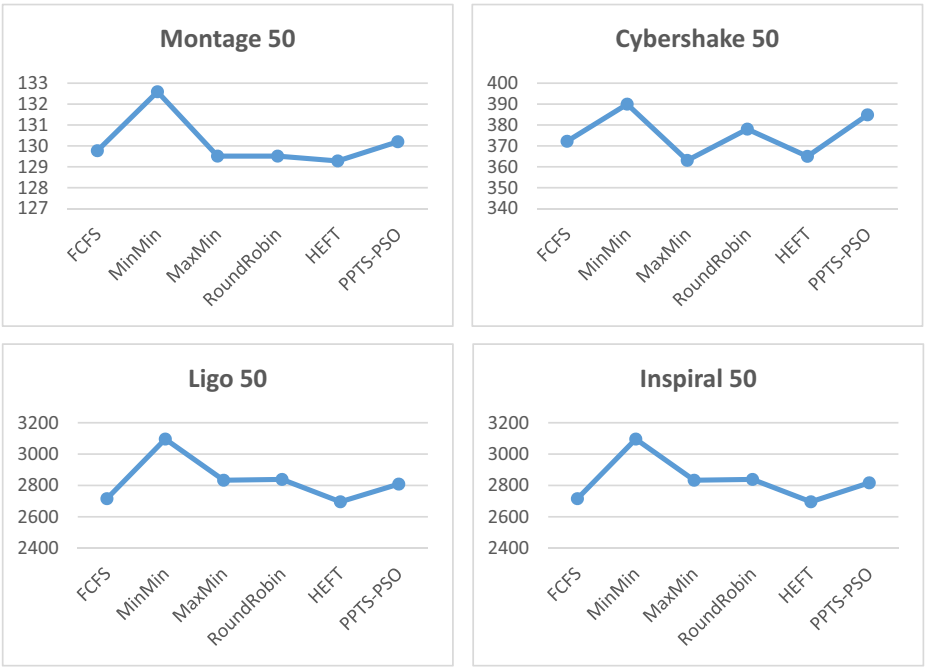

Fig. 7 Simulation results plot of the makespan for 50 tasks and 5 VMs

50(small size), 100 (medium size) and 1000 tasks (large size). For Montage, we discovered that our approach performs better on tasks with a size greater than 50 (Small size). In comparison to HEFT, which executes them in good execution time but with greater cost. PPTS-PSO achieves an execution time of 256,98 and a cost of 3427,37 for a WF of 100 tasks. Our technique outperforms all existing algorithms for WFs with 1000 jobs, with execution

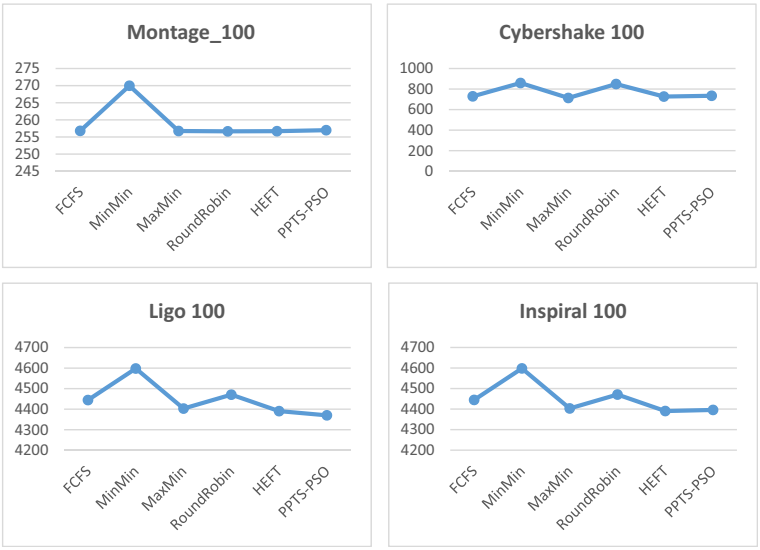

Fig. 8 Simulation results plot of the makespan for 100 tasks and 5 VMs

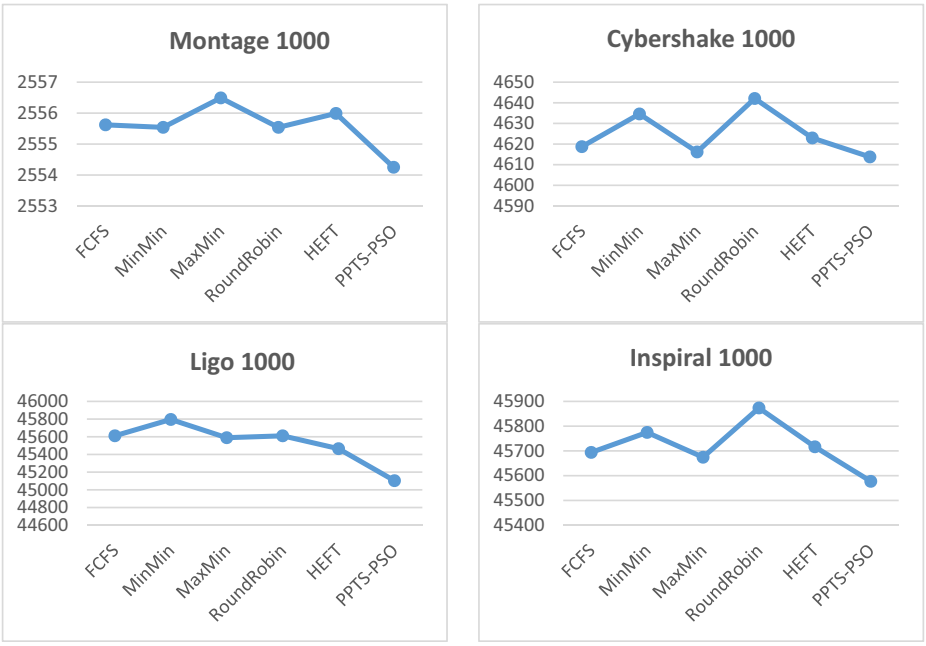

Fig. 9 Simulation results plot of the makespan for 1000 tasks and 5 VMs

times and costs of 2554.25 and 36,002.01, respectively. For cybershake WFs with 50 tasks, our solution has an execution time and cost were (364.84, 38,297.91), which is a better result

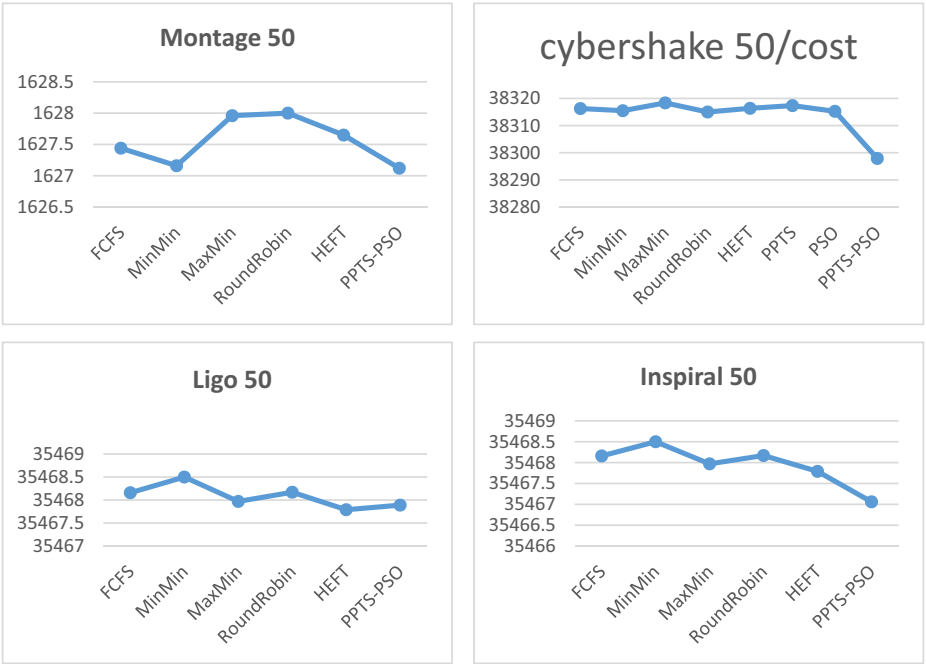

Fig. 10 Simulation results plot of the cost for 50 tasks and 5 VMs

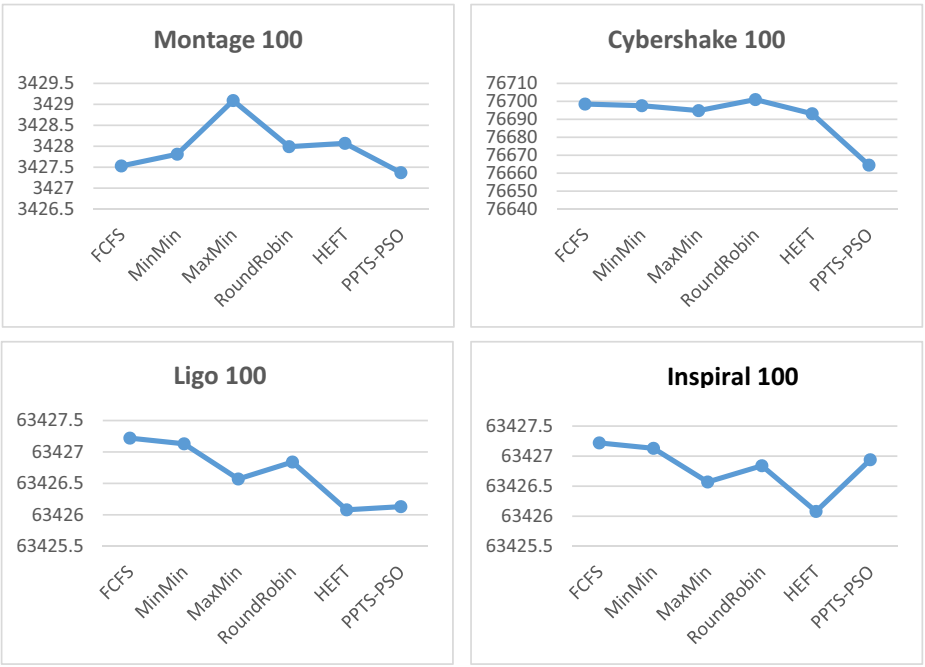

Fig. 11 Simulation results plot of the cost for 100 tasks and 5 VMs

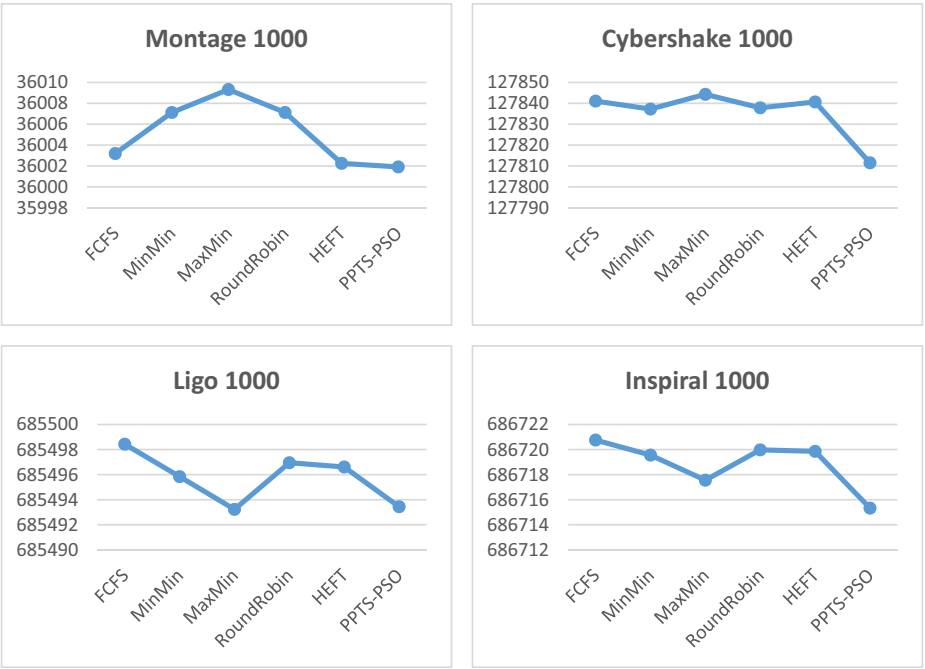

Fig. 12 Simulation results plot of the cost for 1000 tasks and 5 VMs

than HEFT, PPTS, PSO and MAXMIN, which were (365, 38,316.36), (367, 38,317.36), (365.2, 38,315.25), (363.07, 38,318.36) respectively. When it comes to WFs with 100 and 1000, our algorithm outperforms the other heuristic schemes in terms of execution time and cost.

In contrast to Cybershake and Montage, Ligo produced (2808.64, 35,467.89) for WF with 50 tasks in comparison with HEFT whish had good result (2695.84, 35,467.79) and with PPTS which had (2689,26, 35,467,85). for the others sizes 100 and 1000 in terms of both execution time and cost, our solution outcomes the six other approaches with (4370.07, 45,103.43) and (63,426.03, 685,493.22) respectively. The results obtained with Inspiral, for 50 tasks and in comparison with HEFT which has (2695.84, 35,467.79), our solution had the value of time and cost equal to (2816.96, 35,467.06) which is still a better result. For task sizes 100 and 1000 our algorithm has acceptable results (4395.98, 63,426.94) and (45,577.07, 686,715.33) compared to other solutions. On an average and for the makspan, the PPTSPSO algorithm gives 5% improvement with respect to HEFT algorithm, 7% improvement with respect to PSO algorithm and 11.5% improvement with respect to PPTS algorithm. Similarly, for the cost, the algorithm gives 15.6% improvement with respect to HEFT algorithm, 25.5% improvement with respect to PSO algorithm and 17.3% improvement with respect to PSO algorithm. This proves the efficiency of our proposed PPTSPSO algorithm. Finally, we can see that our algorithm produces efficient results for all types of scientific applications tested in our experiments, especially for Ligo and Inspiral applications with sizes of 100 and 1000 tasks, when compared to other solutions that consider the two critical criteria of execution time and cost.

7 Conclusion

This work proposes the PPTS-PSO technique for scheduling tasks from scientific applications in a cloud-computing environment. Our solution beats conventional techniques in terms of overall performance, according to simulation testing utilizing four well-known scientific workflows. The future directions of the proposed solution, is to optimize our algorithm to minimize execution time, increase resource utilization, load balancing and decrease power consumption. We want to address the issue of power consumption in each data center while developing cloud workflows, as well as simulate the Workflows Scheduling in heterogeneous cloud environments that are globally distributed, emphasizing the importance of factoring in data transmission time and cost across data centers. We will concentrate on optimizing more objectives for numerous workflows with different charge models in hybrid cloud environments, as well as refining our PPTSPSO to speed up its convergence speed, particularly when dealing with large-scale and complicated applications.

Data availability statement The data that support the findings of this study are openly available in github, [Online]. Available at: https://github.com/pegasus-isi/pegasus and in our repository at https://github.com/adnanetalha/DAX-file

Declaration The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Adnane talha reports administrative support, equipment, drugs, or supplies, statistical analysis, and writing assistance were provided by Sidi Mohamed Ben Abdellah University. Adnane talha reports a relationship with Sidi Mohamed Ben Abdellah University that includes: non-financial support. Adnane talha has patent pending to adnane talha.

References

1. Abualigah L, Diabat A (2021) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Clust Comput 24(1):205–223. https://doi.org/10.1007/s10586-020-03075-5
2. Ahmad W, Alam B (2021) An efficient list scheduling algorithm with task duplication for scientific big data workflow in heterogeneous computing environments. Concurr Comput Pract Exp 33(5). https://doi.org/10.1002/cpe.5987
3. Ahmad Z, Jehangiri AI, Alaanzy MA, Othman M, Latip R, Zaman SKU, Umar AI (2021) Scientific workflows management and scheduling in cloud computing: taxonomy, prospects, and challenges. IEEE Access 9:53491–53508. https://doi.org/10.1109/ACCESS.2021.3070785
4. Aktan MN, Bulut H (2021) 'Metaheuristic task scheduling algorithms for cloud computing environments', Concurr. Comput. Pract. Exp. https://doi.org/10.1002/cpe.6513
5. Arabnejad H, Barbosa JG (2014) List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table. IEEE Trans PARALLEL Distrib Syst 25(3):13
6. Arif KI (2020) A Hybrid MinMin & Round Robin Approach for task scheduling in cloud computing. Int J Control Autom 13:10
7. Arora N, Kumar R (2020) HPSOGWO: A Hybrid Algorithm for Scientific Workflow Scheduling in Cloud Computing. Int J Adv Comput Sci Appl 11(10). https://doi.org/10.14569/IJACSA.2020.0111078
8. Biswas T, Kuila P, Ray AK (2020) A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems. Clust. Comput 23(4):3255–3271. https://doi.org/10.1007/s10586-020-03085-3
9. Chen W, Deelman E (2012) 'WorkflowSim: a toolkit for simulating scientific workflows in distributed environments', in 2012 IEEE 8th international conference on E-science, Chicago, IL, USA, pp. 1–8. https://doi.org/10.1109/eScience.2012.6404430
10. Chirkin AM et al (2017) Execution time estimation for workflow scheduling. Futur Gener Comput Syst 75:376–387. https://doi.org/10.1016/j.future.2017.01.011
11. Choudhary A, Gupta I, Singh V, Jana PK (2018) A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Futur Gener Comput Syst 83:14–26. https://doi.org/10.1016/j.future.2018.01.005
12. Djigal H, Feng J, Lu J (2019) 'Task scheduling for heterogeneous computing using a predict cost matrix', in proceedings of the 48th international conference on Parallel processing: workshops, Kyoto Japan. pp. 1–10. https://doi.org/10.1145/3339186.3339206
13. Dorigo M, Stützle T (2019) Ant Colony optimization: overview and recent advances. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, vol 272. Springer International Publishing, Cham, pp 311–351. https://doi.org/10.1007/978-3-319-91086-4_10
14. Gharooni-fard G, Moein-darbari F, Deldari H, Morvaridi A (2010) Scheduling of scientific workflows using a chaos-genetic algorithm. Procedia Comput Sci 1(1):1445–1454. https://doi.org/10.1016/j.procs.2010.04.160
15. Haidri RA, Katti CP, Saxena PC (2020) Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. J King Saud Univ - Comput Inf Sci 32(6):666–683. https://doi.org/10.1016/j.jksuci.2017.10.009
16. Hamad SA, Omara FA (2016) Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment. Int J Adv Comput Sci Appl 7(4). https://doi.org/10.14569/IJACSA.2016.070471
17. He Y, Chen J, Du C, and Gu Q (2020) 'Scheduling for heterogeneous computing platforms using a genetic algorithm', in 2020 IEEE 5th information technology and mechatronics engineering conference (ITOEC), Chongqing, China. pp. 1237–1241. https://doi.org/10.1109/ITOEC49072.2020.9141576.
18. Hu Y, Li J, He L (2020) A reformed task scheduling algorithm for heterogeneous distributed systems with energy consumption constraints. Neural Comput Applic 32(10):5681–5693. https://doi.org/10.1007/s00521-019-04415-2
19. Karaboga D, Basturk B (2007) 'Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems', in Foundations of Fuzzy Logic and Soft Computing, vol. 4529, P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, and W. Pedrycz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 789–798 https://doi.org/10.1007/978-3-540-72950-1_77.
20. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, vol 4, pp 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
21. Li H, Wang D, Cañizares Abreu JR, Zhao Q, Bonilla Pineda O (2021) 'PSO+LOA: hybrid constrained optimization for scheduling scientific workflows in the cloud', J. Supercomput. https://doi.org/10.1007/s11227-021-03755-y

22. Liaqat M, Naveed A, Ali RL, Shuja J, Ko K-M (2019) Characterizing dynamic load balancing in cloud environments using virtual machine deployment models. IEEE Access 7:145767–145776. https://doi.org/10.1109/ACCESS.2019.2945499
23. Madhura R, Elizabeth BL, Uthariaraj VR (2021) An improved list-based task scheduling algorithm for fog computing environment. Computing 103(7):1353–1389. https://doi.org/10.1007/s00607-021-00935-9
24. Nishiyama Y, Gunji Y-P, Adamatzky A (2013) Collision-based computing implemented by soldier crab swarms. Int J Parallel Emergent Distrib Syst 28(1):67–74. https://doi.org/10.1080/17445760.2012.662682
25. Orr M, Sinnen O (2020) Integrating task duplication in optimal task scheduling with communication delays. IEEE Trans Parallel Distrib Syst 31(10):2277–2288. https://doi.org/10.1109/TPDS.2020.2989767
26. pegasus, Workflow management system (2018). [Online]. Available: https://pegasus.isi.edu/
27. Polepally V, Shahu Chatrapati K (2019) Dragonfly optimization and constraint measure-based load balancing in cloud computing. Clust Comput 22(S1):1099–1111. https://doi.org/10.1007/s10586-017-1056-4
28. Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
29. Rizvi N, Dharavath R, Edla DR (2021) Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization. Simul Model Pract Theory 110:102328. https://doi.org/10.1016/j.simpat.2021.102328
30. Samadi Y, Zbakh M, Tadonki C (2018) 'E-HEFT: enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing', in 2018 international conference on High Performance Computing & Simulation (HPCS), Orleans. pp. 601–609. https://doi.org/10.1109/HPCS.2018.00100
31. Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N (2015) FUGE: a joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Clust Comput 18(2):829–844. https://doi.org/10.1007/s10586-014-0420-x
32. Taheri G, Khonsari A, Entezari-Maleki R, Sousa L (2020) A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems. Appl Soft Comput 91:106202. https://doi.org/10.1016/j.asoc.2020.106202
33. Topcuoglu H, Hariri S, Wu M-Y (2002) Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Trans PARALLEL Distrib Syst 13(3):15
34. uz Zaman SK, Maqsood T, Ali M, Bilal K, Madani SA, ur Rehman Khan A (2019) A Load Balanced Task Scheduling Heuristic for Large-Scale Computing Systems. Comput Syst Sci Eng 34(2):79–90. https://doi.org/10.32604/csse.2019.34.079
35. Velliangiri S, Karthikeyan P, Arul Xavier VM, Baswaraj D (2021) Hybrid electro search with genetic algorithm for task scheduling in cloud computing. Ain Shams Eng J 12(1):631–639. https://doi.org/10.1016/j.asej.2020.07.003
36. Yuan G-N, Zhang L-N, Liu L-Q, Wang K (2014) Passengers' evacuation in ships based on neighborhood particle swarm optimization. Math Probl Eng 2014:1–10. https://doi.org/10.1155/2014/939723