IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 27, 2019, accepted December 14, 2019, date of publication December 20, 2019,
date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2961106

A Hybrid Algorithm for Scheduling Scientific Workflows in Cloud Computing

MUHAMMAD SARDARAZ AND MUHAMMAD TAHIR
Department of Computer Science, COMSATS University Islamabad at Attock Campus, Attock 43600, Pakistan
Corresponding author: Muhammad Tahir (m_tahir@cuiatk.edu.pk)

ABSTRACT Cloud computing has become the main source for executing scientific experiments. It is an effective technique for distributing and processing tasks on virtual machines. Scientific workflows are complex and demand efficient utilization of cloud resources. Scheduling of scientific workflows is considered as NP-complete. The problem is constrained by some parameters such as Quality of Service (QoS), dependencies between tasks and users' deadlines, etc. There exists a strong literature on scheduling scientific workflows in cloud environments. Solutions include standard schedulers, evolutionary optimization techniques, etc. This article presents a hybrid algorithm for scheduling scientific workflows in cloud environments. In the first phase, the algorithm prepares tasks lists for PSO algorithm. Bottleneck tasks are processed on high priority to reduce execution time. In the next phase, tasks are scheduled with the PSO algorithm to reduce both execution time and monetary cost. The algorithm also monitors the load balance to efficiently utilize cloud resources. Benchmark scientific workflows are used to evaluate the proposed algorithm. The proposed algorithm is compared with standard PSO and specialized schedulers to validate the performance. The results show improvement in execution time, monetary cost without affecting the load balance as compared to other techniques.

INDEX TERMS Cloud computing, PSO, scheduling, scientific workflows.

I. INTRODUCTION

Cloudcomputing has shifted computing from traditional way to a fascinating and impressive era of computing. As opposed to traditional computing, where users need to maintain in-house infrastructure, cloud computing avoids this requirement and provides services as per user demand and use. The users do not need to purchase and maintain hardware, storage and processing, etc. The data are stored and accessed via the Internet [1], [2]. Cloud users can access their data anytime from anywhere. Cloud systems usually deliver three deployment models i.e. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [3]. Cloud computing environment is flexible and scalable that facilitates the users to lease or release services as per need. Users can lease services with long term reservation plans or short term dynamic plans [4]. Many cloud services providers provide similar services such as computing, storage and network services. The services are influenced by non-functional Quality of Service (QoS) parameters such as availability, time, cost and energy consumption, etc., [5]. Cloud computing is deployed in three models i.e. public, private or hybrid mode [6]. In public mode, cloud services are accessible to anyone. In private mode, services can be accessed only by authorized users. Hybrid cloud provides services as mixture of public and private clouds [7].

Growth in data generated in different scientific disciplines demands huge processing power and storage space. Next Generation Sequencing (NGS) machines have produced a huge amount of data in the last two decades that has outpaced the progress in the development of computer hardware [8], [9]. Other scientific fields such as Astronomy, Environmental Sciences, Meteorology, Geological Sciences also produce a large amount of data [10]. To reduce the complexities in the execution of such applications, Workflow Management Systems (WMS) are used to handle large scale data and experiments [11].

Workflows are represented as Directed Acyclic Graphs (DAGs) consisting of n tasks, where vertices represent tasks and edges show the dependencies between the tasks. The number of tasks in scientific workflows is very large. In addition, these jobs have dependencies that make it difficult for

The associate editor coordinating the review of this manuscript and approving it for publication was Xiao Liu.

the scheduler to efficiently schedule tasks and utilize cloud resources. Scheduler acts as an intermediary between cloud resources and workflow tasks. Workflow scheduling in cloud environment is considered as NP-complete. Many factors such as QoS, user deadlines, monetary cost, execution time, data privacy and security, etc. influence the performance of scheduling algorithms. Workflow scheduling algorithms need massive computational resources that make it suitable for cloud computing environments. Cloud computing is economical and scalable infrastructure to execute workflow tasks. Tasks in workflows have different execution times with different computational demands [12]. Some workflows demand high processing power whereas others need large memory and high bandwidth.

Workflow scheduling has been widely studied in the literature. Some researchers have used traditional scheduling algorithms, whereas others have focused on optimization techniques to solve the problem. Solutions include single objective, bi-objective or multi-objective. Most of the researchers have targeted makespan, cost and load balance, etc. Solutions include either heuristics or meta-heuristics. Heuristics like min-min, max-min, etc. or combinations of these techniques with meta-heuristics have been used in literature [5], [13], [14]. Solutions in meta-heuristics use nature-inspired algorithm such as Genetic Algorithm (GA) [15]–[18], Ant Colony Optimization (ACO) [19]–[22], Particle Swarm Optimization (PSO) [23]–[26], etc. Each of the solutions considers specific aspects of workflow scheduling such as tasks dependencies, heterogeneity of resources, scalability, etc., [12].

This article presents a hybrid algorithm for scheduling scientific workflows in cloud computing. The proposed algorithm is based on PSO technique with preprocessing phase to prepare tasks for PSO algorithm. The contribution of the article includes the preprocessing phase before applying PSO. Workflow tasks have dependencies which makes it difficult for the scheduler to obtain an optimum schedule. The proposed algorithm targets the dependencies to obtain better scheduling. The article is organized into the following sections. Related work is presented in section II with discussion on the strengths and weaknesses of each method. Materials and methods are presented in section III followed by results and discussion in section IV. Finally, section V concludes the article.

II. RELATED WORK

Scheduling of scientific workflows in cloud environments is a challenging issue and considered as NP-complete problem [27]. The problem becomes more complicated when tasks are dependent on each other. The scheduler needs to execute tasks with the constraints of dependencies. Heuristics like Max-Min, Min-Min, and Minimum Completion Time, etc. have been used for this purpose [5], [14]. These solutions suffer from the problems of long waiting times and inefficient resources utilization. Hybrid versions of heuristics and meta-heuristics are also available in literature. Kumar and Verma [13] combined min-min, max-min and GA for scheduling. The technique produced better results; however, the time complexity of the algorithm is high. Solutions in meta-heuristics have also been used to solve the problem. These include GA [15]–[18], ACO [19]–[22], [28], and PSO [23]–[26], etc.

This section presents discussion on some meta-heuristics, the more related work to the proposed algorithm. For further details and other methods, readers are referred to review articles [4], [7], [29]–[33] on workflow scheduling.

The literature of scheduling workflows in cloud computing mainly consists of solutions considering execution time with constraints on other parameters i.e. users' budgets, deadlines, cost, load balance, energy consumption, and fault recovery, etc. An algorithm based on PSO for workflow scheduling in cloud computing environment is presented in [24]. The algorithm reduces makespan or cost or any level in between. The algorithm defines the tunable objective and PSO is used to optimally map resources. Authors also proposed heuristics to find bottleneck tasks to reduce makespan in cloud computing. Different algorithms and parameters are used to validate the performance of the proposed algorithm. Comparatively better results are achieved in terms of targeted parameters. The heuristics take long time to select optimal values. Catfish PSO scheduling algorithm for scientific workflows is presented in [25]. The algorithm targets to reduce both makespan and execution cost. The algorithm is simulated and comparative results on benchmark scientific workflows are presented to validate the performance. The algorithm achieved improvements in makespan; however, there is no significant improvement in cost. Another algorithm based on discrete PSO for scheduling in cloud computing is presented in [23]. Authors consider security, completion time, cost, load balancing as objective parameters. The algorithm formulates an initial group based on the speed and position of the particles and finds an optimal solution to adjust the position of particles. Simulation results are shown to validate the performance of the algorithm. The technique improves some parameters on the cost of others. Correlation among parameters is not validated. Authors in Ref. [26] proposed a PSO based budget constrained scheduling algorithm for workflows in cloud computing environment. Authors have focused to minimize makespan while keeping the constraint of users' budgets. The algorithm is evaluated with simulation results to validate its performance. The algorithm takes long time to find the best solution. Another algorithm named as Bi-criteria Priority-based Particle Swarm Optimization is proposed in [34]. The algorithm targets execution time and cost in the presence of users' budget and deadline constraints. Comparative results are presented to validate the algorithm. The proposed algorithm achieves significant improvement in execution time and monetary cost. The algorithm does not consider load among different resources. Another algorithm based on PSO is presented in [35]. The algorithm targets efficient distribution of resources. Monetary cost and makespan are optimization parameters. Experimental results

on benchmark workflows are shown to validate the results. The time complexity of the algorithm is higher as compared to other tools. Another technique that uses GA and PSO in hybrid mode for workflow scheduling [36]. The objectives of the algorithm are execution time, cost and load balance. The algorithm is designed for heterogeneous cloud environments. GA is used to generate the initial population which is further processed with PSO. Experimental results are shown to validate the performance of the algorithm. The technique suffers from longer execution time due to the use of both GA and PSO.

Some researchers have focused on cost as the primary parameter for scheduling workflows in cloud environments. Other parameters targeted with cost includes users' budget and deadlines, etc. A heuristic for workflow scheduling in cloud computing is proposed in [37]. The heuristic is based on PSO and considers both computation and transmission costs while scheduling workflows. Scheduling heuristic and PSO are merged to optimally schedule workflows and reduce cost. The heuristic maps all tasks in the workflow regardless of the dependencies. Ready tasks are then assigned to the resources according to the mapping specified by PSO. The scheduler then waits for ready tasks and the process is repeated. Simulation results are shown to validate the performance of the proposed heuristics. The algorithm achieved better results in comparison to other algorithms. The accuracy of the results cannot be validated due to fast convergence and the problem of local optima. A scheduling algorithm based on modified PSO for scheduling tasks in cloud computing environment is presented in [38]. The algorithm uses fitness function for both resource usage cost and execution cost. The algorithm selects particles with the highest fitness value and updates particles according to previous and global best positions. After optimizing positions and velocity of particles optimization step takes place. The algorithm is compared with standard PSO to validate the performance of the algorithm. The algorithm achieves improvements against standard techniques; however, comparative results with specialized schedulers are not presented. Authors in [39] present a discrete PSO algorithm for scheduling workflows in cloud computing. The algorithm takes into account both data transmission and computation cost. The algorithm starts with an initialization step, where greedy randomized adaptive search procedure is used. In the next step, the best parameters are selected for particles and the best position is returned. Authors present comparative analysis to validate the performance of their algorithm. The proposed technique is not efficient in large search space. A chaotic PSO based task scheduling algorithm in cloud workflows is presented in [40]. The algorithm applies chaotic sequence and adaptive inertia weight factors for task-level scheduling. The prior parameter is used to assure global convergence and the later is used to avoid premature convergence. The main focus has been on the cost. Comparative simulation results with standard PSO and other algorithms are presented to evaluate the proposed algorithm. The proposed technique is not suitable for tasks with large number of dependencies. A multi-objective scheduling algorithm for multi-cloud environment is presented in [41]. The objective parameters of the algorithm are makespan and cost in the presence of reliability constraint. The proposed technique is based on PSO. Initially, the algorithm generates particles with random positions and velocities and applies PSO to generate non-dominated solutions. A coding strategy is applied to generate schedules. Finally, constraints are enforced to ensure that positions of the particles do not move beyond the boundary of the constraints. Comparative results are presented to validate the performance of the proposed technique. Energy consumption has also been used as objective along with other parameters in scheduling algorithms. A multi-objective algorithm for workflow scheduling in cloud computing is presented in [6]. The algorithm is based on PSO and the main focus has been on energy consumption as opposed to other researchers who have focused on makespan, cost or user deadlines. The algorithm uses Dynamic Voltage and Frequency Scaling (DVFS) technique i.e. processors operate at different levels of the voltage supply. Different levels of voltage establish a correlation between quality of scheduling and energy consumption. The algorithm is validated with simulation results and comparative results are presented. The algorithm shows reductions in energy; however, other scheduling parameters are affected.

In literature, researchers have focused on some specific parameters. Some articles have focused on single objectives whereas others have focused on bi or multi-objective. There is a correlation between different parameters i.e. improving one parameter affects other parameters. In most of the cases, multi-objective optimization algorithms have targeted execution time and cost. Other parameters such as load balance and energy consumption, etc. are infew in few studies. In this article, we present an algorithm that targets execution time, cost and load balance as multi-objective optimization parameters.

III. MATERIALS AND METHODS

In this section, we present multi-objective hybrid algorithm for scheduling workflows. The proposed algorithm includes preprocessing, and PSO based scheduling. Following subsections presents details of the algorithm.

A. WORKFLOW AND CLOUD MODEL

Workflows are represented as DAGs. Workflow tasks have dependencies represented as G = (V, E), where V refers to the vertices and represents tasks in workflow and E refers to edges and represents dependencies between tasks. Parent task should be executed before any child task starts execution [42]. Workflow tasks have different parameters such as execution time, data to be sent or received and dependencies between parent-child tasks. Workflow tasks may be computationally intensive, data-intensive or both computation and data intensive.

The cloud model consists of data centers. Each datacenter consists of a number of physical machines. The machines consist of computing and storage resources. Each resource

has processing, storage, memory, and bandwidth capacity. Resources in cloud computing are represented as Virtual Machines (VMs). VMs have fixed bandwidth, processing capabilities and storage cost per unit of time, etc. Workflow tasks can be scheduled for any of the available resources. The processing capacity of VM is measured as the number of Processing Elements (PE) and the processing power of each PE. Equation 1 is used to calculate the processing capacity of a VM.

$$C_i = (PE_{\times} MIPS_i) \tag{1}$$

Equation 2 is used to calculate capacity of n VMs.

$$C = \sum_{i=1}^{n} C_i \tag{2}$$

Load of VM is the ratio of the length of tasks executed by a VM and capacity of VM. Equation 3 is used to calculate the load of VM.

$$L_{vmi} = \frac{TL}{C_i} \tag{3}$$

Load of all VMs can be calculated with Equation 4.

$$L = \sum_{i=1}^{n} L_{vmi} \tag{4}$$

While executing workflows tasks, some parameter needs to be measured. The proposed algorithm uses makespan, cost and load balance as multi-objective optimization parameters.

Completion Time CT of a workflow tasks t_i is the time of getting data and the execution time. The task may need access to the required data for execution. Completion time of task t_i can be calculated with equation 5.

$$Time(t_i) = Time((Trans_{t_i}, t_j) + Time_E(t_i, VM_k)) \tag{5}$$

Where $Trans_{t_i}, t_j$ is the time of data transmission from task t_i to t_j and $Time_E(t_i, VM_k)$ is the execution time of task t_i over VM_k. Transfer time can be calculated with equation 6 whereas execution time can be calculated with equation 7.

$$Trans(t_i, t_j) = \frac{sizeof(t_i, t_j)}{\beta(VM_k, VM_m)} \tag{6}$$

In above equation, $sizeof(t_i, t_j)$ is the size of data transferred from task t_i to t_j and $\beta(VM_k, VM_m)$ is the bandwidth of data centers where VM_k and VM_m are located. In case where both VMs are in same data center, transmission cost will be zero.

$$T_E = \frac{l_i}{C_{m_j}} \tag{7}$$

In equation 7, l_i is the length of tasks i and C_{m_j} is the processing capacity of VM_j calculated with equation 2. Makespan is the finish time of the last task in the workflow. Equation 8 is used to calculate the makespan.

$$MS = FT^{n}_{i=1}[task_i time] \tag{8}$$

Where MS refers to makespan and FT is the finish time of a task. Monetary Cost (MC) of a workflow is the execution cost and data transfer cost between different tasks. Cost is calculated with Equation 9. Monitory cost consists of Total Execution Cost (TEC) calculated with equation 10 and Total Transfer Cost (TTC) calculated with equation 11.

$$MC = TEC + TTC \tag{9}$$

$$TEC_{wi} = \sum_{i=1}^{n} vm_i^{time} \quad vm_i^{cost} \tag{10}$$

where n is the number of tasks in the workflow, vm^{time} refers to the time a VM has executed a particular task. Vm^{cost} is the cost of VM for executing a task.

$$TTC_{wi} = \sum_{i=1}^{n} \frac{size(_{t_i,t_j})}{\beta cost} \tag{11}$$

Size refers to the size of data to be transferred and βcost is the bandwidth cost. Load balance (σ) is the measurement of the standard deviation of the load of all nodes as shown in equation 12. The smaller values mean better load management.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(L_{vmi} - L)^2}{n}} \tag{12}$$

Where L_{vmi} refers to the load of VM_i, L is the average load of all VMs and n is the number of VMs.

B. PARTICLE SWARM OPTIMIZATION (PSO)

PSO is an Evolutionary Algorithm (EA) inspired by the behavior of the Swarm of birds or School of fishes [30]. In a problem space each particle swarms and finds a candidate solution. A particle is represented by two parameters i.e. position p_i and velocity v_i. Position of the particle is influenced by two parameters i.e. $pbest$ and $gbest$. The position $pbest$ is the best position visited by the particle whereas $gbest$ is the best position of the experience of the neighboring particles. After each iteration, the algorithm updates position and velocity. Equation 13 is used to update velocity.

$$v_{id}^{t} = wv_{id}^{t-1} + c1r1(pbest_{id}^{t} - x_{id}^{t}) + c2r2(gbest_{id}^{t} - x_{id}^{t}) \tag{13}$$

where v_{id}^{t} is the velocity for the d_{th} dimension of the i_{th} particle on iteration t. The position is updated according to Equation 14. Where p_{id}^{t} is the position of particle $i$at time t in d_{th} dimension and v_{id}^{t} is the velocity calculated in Equation 13.

$$p_{id}^{t+1} = p_{id}^{t} + v_{id}^{t} \tag{14}$$

Fitness of the solution denoted as F is calculated with Equation 15.

$$F = w\frac{max^{ms} - ms}{max^{ms} - min^{ms}} + w\frac{max^{mc} - mc}{max^{mc} - min^{mc}} + w\frac{max^{lb} - lb}{max^{lb} - min^{lb}} \tag{15}$$

Maximum and minimum values of makespan, cost and load balance are used to calculate the fitness with weightage factor w. In the evaluation of the proposed algorithm, equal weight i.e. 0.33 was used for all parameters.

A multi-objective optimization problem can be formulated with m decision variables and n objectives as shown in equation 16 [6].

$$min(y = f(x) = [f_1(x), \ldots f_n(x)]) \tag{16}$$

where $x = (x_1, \ldots x_m) \in X$ is m dimensional decision vector in search space X and $y = (y_1, \ldots y_n) \in Y$ is the objective vector in objective space Y.

In such problems, no solution can be claimed as optimal with respect to all objectives. Potential candidate solutions can be considered optimal for a set of objectives referred to as Pareto-optimal set. In this work, we target three objectives i.e. makespan, cost and load balance. The target is to minimize all parameter. Keeping in view the minimization criterion, we discuss Pareto concepts related to the problem. Suppose, we have two decision vectors x^1 and x^2. The decision vector x^1 is said to be dominant over x^2 if and only if x^1 is as good as x^2 for all objective and x^1 is strictly superior than x^2 in at least one objective. The definition of Pareto dominance can be written mathematically as in equation 17.

$$x^1 \prec x^2 \iff \forall i f_i(x^1) \leq f_i(x^2) \wedge \exists j f_j(x^1) < f_j(x^2) \tag{17}$$

The decision vector x^1 is said to be Pareto optimal if and only if x^1 is not dominated by any other decision vector as shown in equation 18.

$$\nexists x^2 \in X : x^2 \prec x^1 \tag{18}$$

A set of all Pareto optimal decision vectors is referred to Pareto optimal set as shown in equation 19.

$$P_S = \{x^1 \in X, |\nexists x^2 \in X : x^2 \prec x^1\} \tag{19}$$

The image of Pareto optimal set is referred to as the Pareto optimal front as shown in equation 20.

$$P_F = \{f(x) = (f_1(x), \ldots f_n(x)) | x \in P_S\} \tag{20}$$

Standard PSO algorithm follows the following steps. (1) Initialize the initial population of particles with random passion and velocity.

(2) Evaluate the objective value of each particle. Update *pbest* and *gbest* equal to the current position and best initial particle respectively.

(3) Update velocity and position of each particle.

(4) Evaluate fitness of each particle according to fitness function.

(5) For each particle select *pbest*. If current value is better than *pbestt*, update current value.

(6) From the whole population select *gbest* value. If the obtained value is better than *gbest*, update *gbest* according to position and value.

(7) Repeat the process until the stopping criterion is met.

C. PROPOSED ALGORITHM

The proposed algorithm is based on PSO technique to reduce monetary cost and execution time (makespan) of workflows with a balanced load over all nodes. The main advantage of PSO over other meta-heuristics is speed and faster convergence of the algorithm. The algorithm has been used in the literature of cloud computing for different purposes such as scheduling and VMs placement, etc. It is assumed that the scheduler knows dependencies between various workflow tasks. The execution time of workflow tasks is also known in advance. The purpose of the proposed algorithm is to schedule cloud resources to workflow tasks by optimizing monetary cost and execution time. The goal of the scheduling algorithm is to assign resource R_i to workflow W_j such that cloud resources are efficiently utilized. The scheduler must also consider other parameters while scheduling cloud resources.

Before applying PSO, the proposed algorithm uses preprocessing steps to prepare tasks and resources for PSO. The proposed algorithm first sorts tasks according to the number of descendants i.e. the tasks with large number of descendants are processed first. These jobs become bottleneck for cloud resources and thus cause large execution times [24]. The algorithm also sorts cloud resources according to the processing power i.e. resources with high processing power and low processing power. Two lists of resources are maintained to process workflow tasks. Parent tasks that need large processing time are processed with high processing nodes to eliminate the dependencies quickly. After processing parent tasks, children tasks are processed according to their position in the graph i.e. leaf tasks are processed with low processing nodes, parent and intermediate tasks are processed with available high processing nodes. If at any point in time the resources are not load balanced the algorithm switches tasks from one list to the other to make effective and balanced use of resources. The tasks are moved to dependency list that contains tasks with dependency and independent list that contains independent tasks. The algorithm uses PSO for assignment of resources to tasks for both lists. Tasks start execution with random particle position and velocity and each particle is evaluated according to the fitness function. Variables are updated, and the process is repeated until the stopping criterion does not meet. Fitness is calculated on the basis of execution time and cost as shown in equation 15.

Figure 1 shows the flow of the proposed algorithm, whereas algorithms 1 and 2 show the procedures involved. Lines 1-4 in algorithm 1 are used to separate root tasks. These tasks are stored in a separate list. Lines 5-7 check intermediate tasks and tasks are moved to respective list accordingly. Line 9 checks the status of the parent tasks in order to start the execution of dependent tasks. In line 12 leaf tasks are identified and moved to a separate list.

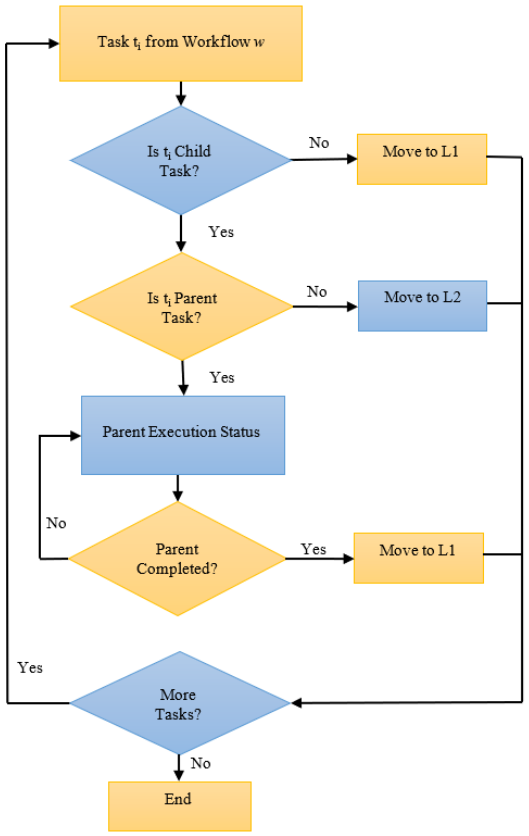

FIGURE 1. Flow of the preprocessing phase of the proposed algorithm.

Algorithm 1 Preprocessing

Input : workflow w
Output : lists of tasks

while w *has tasks* **do**
  if t_i *is not a child task (root task)* **then**
    move t_i to $List_1$
  end
  if t_i *is parent task (intermediate tasks)* **then**
    if *parent tasks of* t_i *are completed* **then**
      move t_i to $List_1$
    else
      wait for parent tasks completion
    end
  else
    move t_i to $List_2$ (leaf tasks)
  end
end
Process both lists with PSO algorithm

Algorithm 2 PSO Based Scheduling Algorithm

Input : List of tasks from pre-processing phase
Output : Tasks to VMs map

N=number of VMs
M=number of tasks in workflow
P=population size
p=i_{th} particle in P
calculate fitness of p according to Eq.15
calculate velocity of p according to Eq.13
$gbest$=global best position
$pbest$=particles' best position
for *each particle p in P* **do**
  for *each task t in workflow* **do**
    l initialize X_{ij}^t randomly
    l initialize velocity v randomly
    evaluate p_i
    update $pbest$ and $gbest$
  end
end

Designing a successful mapping of tasks and resources is a challenging task [43]. We use a search space of m dimensions for m tasks with a set of possible discrete values in the range of 1 to N, where N is the number of VMs. We use notations used in previous study [23] to represent tasks to VMs assignment i.e. $x_i{}^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{ij}^t)$. Where x_{ij}^t represents VM_i assigned by the jth place of a particle at time t. The dimensions of the particle are represented by the number of tasks in a workflow. The velocity is represented by $v_i{}^t = (v_{i1}^t, v_{i2}^t, \ldots, v_{ij}^t)$. Where v_{ij}^t represents the velocity of VM_i to move to jth place of a particle at time t. The proposed algorithm archives feasible solutions i.e. non-dominated particles. Initially the archive is empty. As the algorithm finds a solution, it is stored in the archive. It should be noted that the archive contains only non-dominated solutions. During the process, if the current solution is dominated by another solution, the current solution replaced by the new one in archive. The decision is made on the basis of the fitness criteria used. At the end of the procedure, the archive only contains non-dominated solution referred to as feasible solutions.

IV. EXPERIMENTAL EVALUATION

This section presents experimental setup followed by experimental results and discussion. Scientific workflows [44] from different scientific domains were used for experimental evaluation of the proposed algorithm. Workflows consist of different number of tasks, dependency levels and data transfer between different tasks. Structures of the workflows are shown in Figure 2. Table 1 shows details of the datasets used for experiments. Algorithms were evaluated in terms of makespan, cost and load balance. Makespan refers to the

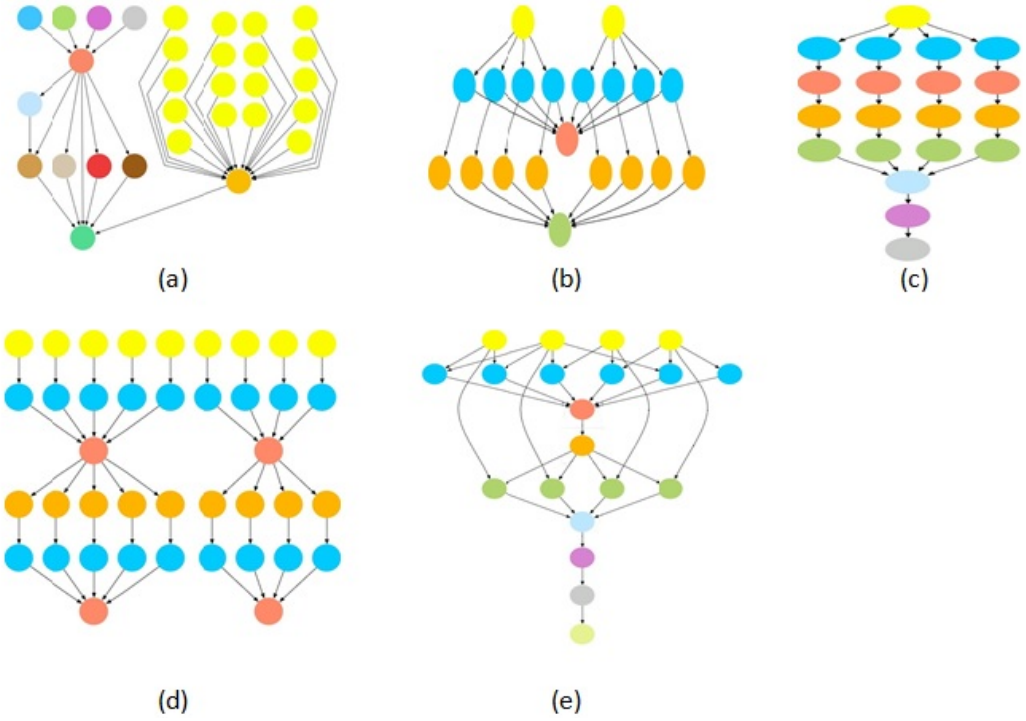

FIGURE 2. Structures of the workflows used for experiments (a) Sipht, (b) CyberShake (c) Epigenomics (d) LIGO (e) Montage.

TABLE 1. Datasets used for experiments.

Dataset	Nodes	w-levels	Parallel Tasks	Average File Size (MBS)	Average Execution Time (s)
Montage	100	9	62	18.05	10.19
Sipht	100	7	51	22.34	173.34
LIGO	100	8	24	28.8	209.78
CyberShake	100	5	48	999.41	21.82
Epigenomics	100	8	24	4033.59	1277.21

total execution time of all tasks in a workflow. Cost refers to execution and data transfer cost of tasks in the workflow application. Load balance is the matrix that shows whether the system is well load balanced. Load balance is measured as the standard deviation of load of all nodes as shown in equation 12. In case of all parameters, small values are preferred.

The algorithm is compared with standard PSO, GA and specialized schedulers based on PSO technique i.e. PSO-DS [35] and GA-PSO [36]. We have used standard version of PSO without any pre-processing phase. The values of parameters, notations and functions used for standard PSO were same as used for the proposed algorithm. A generational standard version of GA was used with roulette wheel selection, multipoint crossover and single point mutation. The values of crossover and mutation probabilities were set to 0.6 and 0.2 respectively. Fitness function and tasks to VMs notations used to evaluate the proposed algorithms were also used for GA. GA-PSO and PSO-DS were simulated with parameters and functions discussed in the respective articles. CloudSim simulator [45] was used for experiments. Experiments were performed on a computer with Intel Core i3 processor and 4 GB memory running Ubuntu 14.04 operating system. The simulation environment consists of 8 VMs with different specifications (1000 MBs memory and MIPS between 1000 and 10000) and 3 datacenters. Costs were set to 0.017, 0.05, 0.01 and 0.01 for processing, memory, storage and transfer respectively. Each algorithm was executed 20 times and average results are shown. Algorithms were evaluated in terms of execution time, cost and load balance. The value of inertia weight factor ω was set to 1.2 and learning factors were set to 2 as recommended in previous studies [35]. Comparative results of different algorithms are shown in Table 2. The results are shown in terms of execution time, cost and load balance. The results show that the proposed algorithm performs better for execution time, cost and load balance on all datasets. On Montage dataset, the proposed algorithm achieves 42.8, 40.9 and 32.2 percent improvement for makespan, cost and load balance respectively over standard PSO algorithm. For the same dataset, percent improvement over GA is 41.04, 35.5 and 45.4 for makespan, cost, load balance respectively. In comparison to GA-PSO, the proposed algorithm achieves 18, 23.5 and 17.1 percent improvement for makespan, cost and load balance respectively. Percent improvement on montage datasets over PSO-DS for makespan, cost, load balance

TABLE 2. Comparative results of proposed algorithm with other scheduling algorithms. Makespan is shown in seconds, cost is shown in dollars and Load Balance (LB) is calculated with Equation 12. Average results of 20 runs are shown.

Parameters	PSO	GA	GA-PSO	PSO-DS	Proposed
Montage					
Makespan	301	292	211	194	172
Cost	1.32	1.21	1.02	0.93	0.78
LB	214	266	175	148	145
Sipht					
Makespan	4839	4747	3637	3482	3387
Cost	1.72	1.68	1.02	0.96	0.82
LB	189	218	184	153	143
LIGO					
Makespan	4862	4782	3613	3510	3449
Cost	1.21	1.32	0.91	0.84	0.67
LB	199	247	192	180	162
CyberShake					
Makespan	6252	6102	4544	4451	4392
Cost	2.26	2.37	1.52	1.36	1.21
LB	212	233	195	182	169
Epigenomics					
Makespan	73557	78332	34399	24781	31756
Cost	6.33	6.55	4.93	4.79	4.21
LB	241	236	247	193	175

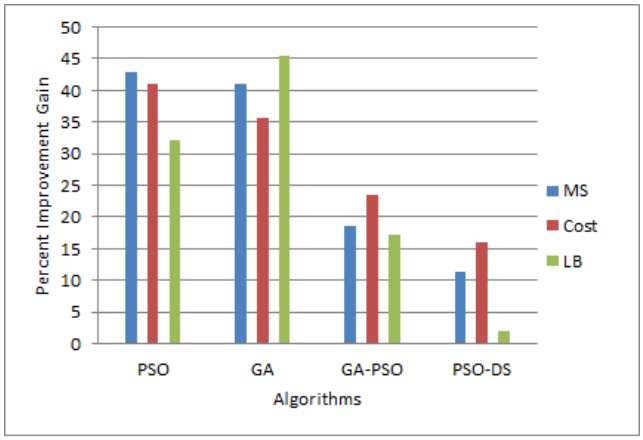

FIGURE 3. Percent improvement gain of the proposed algorithm over PSO, GA, GA-PSO, PSO-DS on Montage dataset.

is 11.3, 16.1 and 2.02 respectively as shown in figure 3. The proposed algorithm achieved 30.0, 52.3 and 24.3 percent improvement in makespan, cost and load balance respectively over standard PSO on Sipht dataset. For the same dataset, the proposed algorithm achieved percent of improvement of 28.6 in makespan, 51.1 in cost and 34.4 in load balance over GA. Percent improvement of the proposed algorithm for Sipht dataset over GA-PSO is 6.8 for makespan, 19.6 for cost and 22.2 for load balance. For the same dataset, the proposed algorithm achieved 2.7, 14.5 and 6.5 percent improvement for makespan, cost and load balance respectively over PSO-DS as shown in figure 4. Percent improvement of the proposed algorithm on LIGO dataset, over standard PSO is 29.06, 44.6 and 18.5 for the three parameters. Improvement in

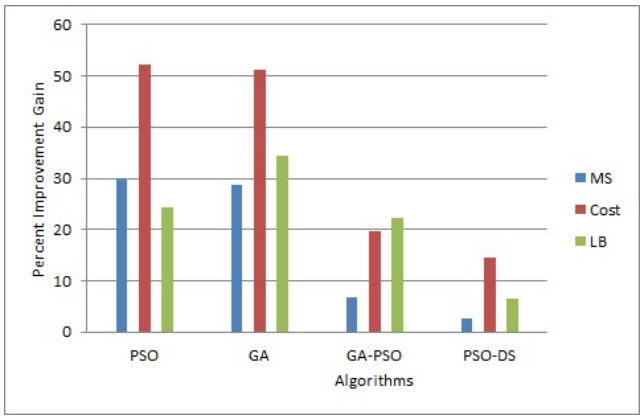

FIGURE 4. Percent improvement gain of the proposed algorithm over PSO, GA, GA-PSO, PSO-DS on Sipht dataset.

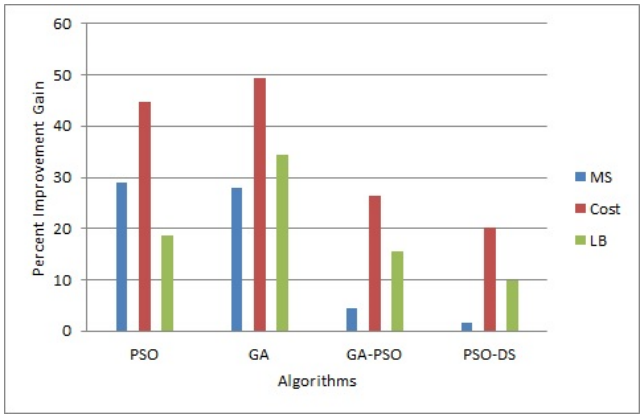

FIGURE 5. Percent improvement gain of the proposed algorithm over PSO, GA, GA-PSO, PSO-DS on Ligo dataset.

execution time, cost and load balance over GA for the same dataset is 27.8, 49.2 and 34.4 respectively. In comparison, GA-PSO, the proposed algorithm achieves 4.5, 26.3 and 15.6 percent improvement for makespan, cost and load balance respectively. In comparison to PSO-DS, the proposed algorithm achieved 1.73, 20.2 and 10 percent improvement for makespan, cost and load balance respectively. Detailed results of LIGO dataset are shown in Figure 5. In case of CyberShake dataset, the proposed algorithm achieved 29.7, 46.4 and 20.2 percent improvement in makespan, cost and load balance respectively over PSO. The improvement gain over GA for the same dataset and parameters is 28.02, 48.9 and 27.4 percent. In comparison to GA-PSO, the proposed algorithm achieved 12.9, 20.3 and 24.8 percent improvement for makespan, cost and load balance respectively. Percent improvement over PSO-DS on CyberShake dataset is 1.3, 11.0 and 7.14 for makespan, cost, load balance respectively. Figure 6 shows the detailed results of CyberShake dataset. On Epigenomics dataset, the proposed algorithm achieved percent improvement of 56.8, 33.4 and 27.3 for makespan, cost and load balance respectively over PSO. For the same dataset, the improvement over GA is 59.4, 35.7 and 25.8 for the three parameters. In comparison to GA-PSO, the proposed algorithm achieved 50.6, 35.5 and

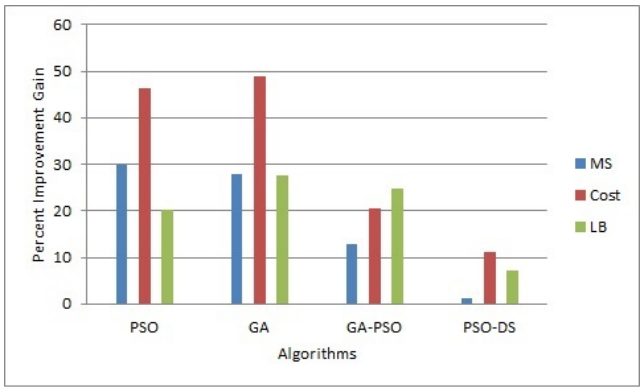

FIGURE 6. Percent improvement gain of the proposed algorithm over PSO, GA, GA-PSO, PSO-DS on CyberShake dataset.

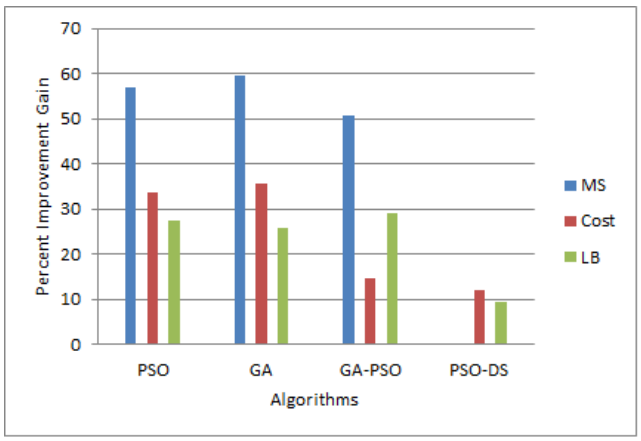

FIGURE 7. Percent improvement gain of the proposed algorithm over PSO, GA, GA-PSO, PSO-DS on Epigenomics dataset.

29.1 percent improvement for makespan, cost and load balance respectively. PSO-DS performed better in case of Epigenomics datasets for makespan parameter. In case of cost and load balance, the proposed algorithm achieved 12.1 and 9.3 percent improvement over PSO-DS. The detailed results of Epigenomics dataset are shown in Figure 7.

The results are influenced by the characteristics of the datasets. For CPU intensive datasets, the proposed algorithm makes efficient utilization of parallelism to reduce execution time, cost and maintain balanced load among nodes. For example, in Epigenomics dataset tasks are grouped into 24 pipelines with 4 nodes in each group. The dependencies are avoided quickly and the execution time and cost are reduced. Better parallelism also results in better load management among resources. Similarly, Sipht dataset has bottleneck tasks i.e. more tasks are dependent on some levels. Avoiding such dependencies results improvement in targeted parameters. In CyberShake dataset, tasks are mainly distributed on two levels. Number of parallel tasks is also high in this dataset. Using small number of dependencies and high parallelism reduces execution time, cost with better load balance. Ligo dataset also has more parallel tasks with few dependencies. Thus, allowing the scheduler to achieve better schedules. Montage dataset consists of large number of parallel tasks with low execution times. Due to high parallelism, better results are achieved.

V. CONCLUSION

This article presents a hybrid scheduling algorithm for scientific workflows in cloud computing environments. The algorithm targets execution time and monetary cost as well as load balance among computing nodes. The algorithm preprocesses the workflow tasks to eliminate bottleneck tasks and further scheduling is performed with the PSO algorithm. The proposed algorithm is validated with experimental results on scientific workflows from different domains. The algorithm achieves improvement over the existing algorithms in targeted parameters. The algorithm is limited to targeted parameters and we aim to include other parameters in the future.

REFERENCES

[1] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, 2013.

[2] Y. Hao, G. Liu, and J. Lu, "Three levels load balancing on CloudSim," *Int. J. Grid Distrib. Comput.*, vol. 7, no. 3, pp. 71–88, 2014.

[3] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments," *J. Comput. Sci.*, vol. 26, pp. 318–331, May 2018.

[4] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of pso-based scheduling algorithms in cloud computing," *J. Netw. Syst. Manage.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.

[5] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, Feb. 2017.

[6] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *Sci. World J.*, vol. 2013, Sep. 2013, Art. no. 350934.

[7] K. Maryam, M. Sardaraz, and M. Tahir, "Evolutionary algorithms in cloud computing from the perspective of energy consumption: A review," in *Proc. 14th Int. Conf. Emerg. Technol. (ICET)*, 2018, pp. 1–6.

[8] S. D. Kahn, "On the future of genomic data," *Science*, vol. 331, no. 6018, pp. 728–729, 2011.

[9] M. Sardaraz, M. Tahir, and A. A. Ikram, "Advances in high throughput dna sequence data compression," *J. Bioinf. Comput. Biol.*, vol. 14, no. 3, 2016, Art. no. 1630002.

[10] S. G. Ahmad, C. S. Liew, E. U. Munir, T. F. Ang, and S. U. Khan, "A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems," *J. Parallel Distrib. Comput.*, vol. 87, pp. 80–90, Jan. 2016.

[11] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing," *ACM Sigmod Rec.*, vol. 34, no. 3, pp. 44–49, 2005.

[12] N. Anwar and H. Deng, "Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments," *Future Internet*, vol. 10, no. 1, p. 5, 2018.

[13] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," in *Proc. Int. Conf. Adv. Comput., Commun. Informat.*, 2012, pp. 137–142.

[14] H. Xu, B. Yang, W. Qi, and E. Ahene, "A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 3, pp. 976–995, Mar. 2016.

[15] E. Barrett, E. Howley, and J. Duggan, "A learning architecture for scheduling workflow applications in the cloud," in *Proc. IEEE 9th Eur. Conf. Web Services (ECOWS)*, Sep. 2011, pp. 83–90.

[16] C. Chen, J. Liu, Y. Wen, J. Chen, and D. Zhou, "A hybrid genetic algorithm for privacy and cost aware scheduling of data intensive workflow in cloud," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.* Cham, Switzerland: Springer, 2015, pp. 216–222.

[17] A. Verma and S. Kaushal, "Budget constrained priority based genetic algorithm for workflow scheduling in cloud," in *Proc. 5th Int. Conf. Adv. Recent Technol. Commun. Comput. (ARTCom)*, 2013, pp. 578–591.
[18] Y. Aryan and A. G. Delavar, "A bi-objective workflow application scheduling in cloud computing systems," *Int. J. Integr. Technol. Educ.*, vol. 3, no. 2, pp. 51–62, 2014.
[19] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *Proc. 8th Int. Conf. Comput. Eng. Syst. (ICCES)*, 2013, pp. 64–69.
[20] Y. Zhaofeng and F. Aiwan, "Application of ant colony algorithm in cloud resource scheduling based on three constraint conditions," in *Proc. 5th Int. Conf. Comput. Sci. Technol.*, Apr. 2016, pp. 22–23.
[21] Y. Zhou and X. Huang, "Scheduling workflow in cloud computing based on ant colony optimization algorithm," in *Proc. 6th Int. Conf. Bus. Intell. Financial Eng. (BIFE)*, 2013, pp. 57–61.
[22] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.
[23] C. Jianfang, C. Junjie, and Z. Qingshan, "An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm," *Cybern. Inf. Technol.*, vol. 14, no. 1, pp. 25–39, 2014.
[24] K. Wu, "A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing," San Jose State Univ., San Jose, CA, USA, Tech. Rep. 358, 2014.
[25] S. J. Nirmala and S. M. S. Bhanu, "Catfish-PSO based scheduling of scientific workflows in IaaS cloud," *Computing*, vol. 98, no. 11, pp. 1091–1109, 2016.
[26] X. Wang, B. Cao, C. Hou, L. Xiong, and J. Fan, "Scheduling budget constrained cloud workflows with particle swarm optimization," in *Proc. IEEE Conf. Collaboration Internet Comput. (CIC)*, Oct. 2015, pp. 219–226.
[27] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*. Berlin, Germany: Springer, 2008, pp. 173–214.
[28] Y. Zhaofeng and F. Aiwan, "Application of ant colony algorithm in cloud resource scheduling based on three constraint conditions," *Adv. Sci. Technol. Lett.*, vol. 123, no. 7, pp. 215–219, 2016.
[29] L. Liu, M. Zhang, Y. Lin, and L. Qin, "A survey on workflow management and scheduling in cloud computing," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2014, pp. 837–846.
[30] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *J. Netw. Comput. Appl.*, vol. 66, pp. 64–82, May 2016.
[31] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 8, p. e4041, 2017.
[32] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, Mar. 2014.
[33] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, May 2010.
[34] A. Verma and S. Kaushal, "Cost-time efficient scheduling plan for executing workflows in the cloud," *J. Grid Comput.*, vol. 13, no. 4, pp. 495–506, Dec. 2015.
[35] I. Casas, J. Taheri, R. Ranjan, and A. Y. Zomaya, "PSO-DS: A scheduling engine for scientific workflow managers," *J. Supercomput.*, vol. 73, no. 9, pp. 3924–3947, 2017.
[36] A. M. Manasrah and H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, Jan. 2018, Art. no. 1934784.
[37] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 400–407.
[38] G. Zhao, "Cost-aware scheduling algorithm based on pso in cloud computing environment," *Int. J. Grid Distrib. Comput.*, vol. 7, no. 1, pp. 33–42, 2014.
[39] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *Proc. Int. Conf. Comput. Intell. Secur. (CIS)*, 2010, pp. 184–188.
[40] X. Li, J. Xu, and Y. Yang, "A chaotic particle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems," *Comput. Intell. Neurosci.*, vol. 2015, Jan. 2015, Art. no. 81.
[41] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *J. Netw. Comput. Appl.*, vol. 114, pp. 108–122, Jul. 2018.
[42] L. Liu, M. Zhang, R. Buyya, and Q. Fan, "Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 5, p. e3942, 2017.
[43] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocess. Microsyst.*, vol. 26, no. 8, pp. 363–371, 2002.
[44] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci.*, 2008, pp. 1–10.
[45] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

MUHAMMAD SARDARAZ received the master's degree in computer science from Foundation University Islamabad and the Ph.D. degree in computer science from Iqra University Islamabad, Pakistan, in 2016. He worked as a Lecturer with the Department of Computer Science, University of Wah, Wah Cantt. He is currently working as an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad, Attock, Pakistan. His research interests are cloud computing, cluster and grid computing, and bioinformatics.

MUHAMMAD TAHIR received the Ph.D. degree in computer science from the Department of Computing and Technology, Iqra University, in 2016. He worked as a Lecturer with the Department of Computer Science, University of Wah, Wah Cantt. He is currently working as an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad, Attock Campus. His research interests include parallel and distributed computing, Hadoop MapReduce framework, bioinformatics algorithms design and analysis and sequence alignment, and cloud computing.