Received August 12, 2019, accepted August 31, 2019, date of publication September 3, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939294

A Hybrid Algorithm for Multi-Objective Scientific Workflow Scheduling in IaaS Cloud

YONGQIANG GAO, SHUYUN ZHANG, AND JIANTAO ZHOU
College of Computer Science, Inner Mongolia University, Hohhot 010021, China

Corresponding author: Yongqiang Gao (gaoyongqiang@imu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61662052, in part by the Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering, and in part by the Inner Mongolia Application Technology Research and Development Funding Project.

ABSTRACT With the increase in deployment of scientific workflow applications on an IaaS cloud computing environment, the distribution of workflow tasks to particular cloud instances to decrease run-time and cost has emerged as an important challenge. The cloud workflow scheduling is a well-known NP-hard problem. In this paper, we propose a new approach for multi-objective workflow scheduling in IaaS clouds offering a limited amount of instances and a flexible combination of instance types, and present a hybrid algorithm combining genetic algorithm, artificial bee colony optimization and decoding heuristic for scheduling workflow tasks over the available cloud resources while trying to optimize the workflow makespan and cost simultaneously. The proposed algorithm is evaluated for real-world scientific applications by a simulation process. The simulation results show that our proposed scheduling algorithm performs better than the current state-of-the-art algorithms. We validate the results by the Wilcoxon signed-rank test.

INDEX TERMS Cloud computing, multi-objective optimization, workflow schedule.

I. INTRODUCTION

Scientific workflow is a widely-used model to describe various scientific computing problems in areas such as bioinformatics, astronomy, and physics. With the ever-growing complexity of scientific computing systems, scientific workflows are becoming increasingly data-intensive, communication-intensive and computation-intensive. Generally, these scientific workflows are very large in size since they consist of many independent and/or dependent tasks and thus they require huge infrastructure for their computation, communication, and storage. The scheduling of these tasks on distributed resources has been studied extensively over the years, focusing on heterogeneous environments like distributed computing, grid computing and parallel computing. However, with the emergence of cloud computing, there are many new challenges that must be addressed in order to efficiently schedule the large scale scientific workflow application in cloud environment. This is because that the cloud environment significantly differs from the traditional heterogeneous environments in terms of the computing, data and pricing models.

As a new form of computing service, cloud computing enables the delivery of the resources and services over the Internet, and follows a pay-as-you-go model where consumers are charged according to the use they make of the service. Infrastructure as a Service (IaaS) is the most common type of cloud service, providing users with the ability to lease or release pre-configured virtual machines (VMs) from a cloud infrastructure. Using the VMs, which are called instances in IaaS clouds, users can access to a resource pool with much lower ownership cost for executing applications. Workflow scheduling in IaaS clouds, which is known to be NP-complete, is basically to find proper schemes of assigning tasks to VMs so as to satisfy some performance criteria, such as the overall completion time and the overall execution cost. Many meta-heuristic algorithms, such as genetic algorithm (GA) [1] particle swarm optimization (PSO) [2], and artificial bee colony algorithm (ABC) [3] were proposed to solve the scheduling problem of the workflow tasks in cloud environments.

Standard ABC has great exploitation ability and fast convergence speed, whereas traditional GA has effective

The associate editor coordinating the review of this article and approving it for publication was Jun Huang.

exploration ability. When being applied for solving workflow scheduling problems, ABC can easily get stuck in the local optimal because of its low global exploration efficiency; GA has slower convergence speed in some cases because of the lack of powerful local exploitation ability. To overcome the disadvantages of the two algorithms, this paper proposes a hybrid approach based on GA and ABC for scheduling scientific workflows in IaaS clouds with pay-per-use pricing model. A decoding heuristic is integrated into the hybrid approach to generate a feasible schedule. The performance of the proposed algorithm is evaluated against other algorithms to prove its effectiveness in solving the workflow scheduling problem in IaaS clouds. The contributions of this paper can be summarized as follows:

- We develop a new Pareto-based multi-objective workflow scheduling algorithm, called HGAABC, for optimizing the workflow makespan and cost simultaneously. We extend HGAABC for coping with commercial IaaS cloud computing systems providing a limited amount of instances and a flexible combination of instance types.
- We integrate the ability of exploitation in ABC with the ability of exploration in GA to map each task into a corresponding VM type of instance series in terms of pay-per-use pricing model. A decoding heuristic is proposed to generate a feasible schedule.
- We conduct extensive simulation experiments to evaluate the effectiveness of the proposed HGAABC. We compare our HGAABC with a number of state-of-the-art multi-objective optimization algorithms, including HPSO [4] and PBACO [5]. Extensive experimental results on real-world scientific applications show the efficacy of our algorithm. We validate the results by the Wilcoxon signed-rank test.

The remaining sections of the paper are organized as follows. A survey on related works is presented in section II. Section III presents the models and problem formulation. Section IV describes the proposed HGAABC in detail. The experimental results are given in Section V. Finally, the conclusion is drawn in Section VI.

II. RELATED WORK

Scheduling of workflows is an NP-complete problem [6]. Many heuristic algorithms are developed to solve the tasks scheduling problem using different strategies. Heterogeneous Earliest Finish Time (HEFT) [7] is a popular heuristic which was initially designed for task scheduling in heterogeneous multiprocessor systems. An extension of HEFT called MOHEFT was proposed by Durillo and Prodan [8] for workflow scheduling in Amazon EC2, to minimize makespan and cost of execution. Wu *et al.* [9] proposed a heuristic algorithm, called Critical-Greedy, to solve the budget-constrained workflow scheduling problem for delay minimization in cloud computing environments. Xu *et al.* [10] developed a heuristic-based algorithm called Min-min based time and cost tradeoff (MTCT) to solve the problem of workflow scheduling in clouds considering fault recovery, makespan and cost. Rimal and Maier [11] proposed a cloud-based workflow scheduling (CWSA) policy for compute-intensive workflow applications in multi-tenant cloud computing environments, which aims to minimize the scheduling execution time, workflow execution costs, and total expected tardiness. Deldari *et al.* [12] proposed a cluster combining algorithm (CCA) for workflow scheduling on the multicore cloud, which aims to minimize the monetary costs of workflow execution while meeting a user-defined deadline.

Meanwhile, a number of nature-inspired metaheuristic-based algorithms have been applied in solving the task scheduling problem. Rodriguez and Buyya [2] proposed a particle swarm optimization (PSO) based algorithm for scheduling a scientific workflow application on IaaS clouds, which aims to minimize the overall workflow execution cost while meeting deadline constraints. Liu *et al.* [13] proposed an coevolutionary genetic algorithm (CGA) with adaptive penalty function to find a proper task-VM mapping strategy, which aims to minimizes the total financial cost and the makespan under the deadline constraint. An evolutionary multi-objective optimization (EMO)-based algorithm [14] was proposed to solve the workflow scheduling problem which optimizes both makespan and cost for the cloud environments. Li *et al.* [15] proposed a PSO-based task scheduling algorithm for scientific workflow in clouds to minimize the total workflow execution cost while meeting the deadline and risk rate constraints. Rehman *et al.* [1] proposed a multi-objective genetic algorithm for the scheduling of scientific workflows in cloud environments, which aims to minimize the energy consumption and the makespan under the budget and deadline constraint.

Recently, some hybrid algorithms are proposed to solve multi-objective workflow scheduling problems in the cloud. Anwar and Deng [16] designed a hybrid metaheuristic based on predict earliest finish time (PEFT) and symbiotic organisms search (SOS) algorithms for the workflow scheduling problem in the cloud, which aims to minimize the makespan, the execution cost, and the inefficient utilization of the VMs. Manasrah and Ali [17] presented a hybrid GA-PSO algorithm for cloud scientific workflow scheduling, which aims to reduce the makespan and the cost and balance the load of the dependent tasks over the heterogonous VMs in clouds. Choudhary *et al.* [18] presented a hybrid algorithm based on HEFT and gravitational search algorithm (GSA) for workflow scheduling that considers minimization of makespan and cost. Verma and Kaushal [4] proposed the multi-objective hybrid particle swarm optimization (HPSO) algorithm based upon non-dominance sorting procedure to give a trade-off schedule plan between cost and makespan depending upon the user preference for scientific workflow scheduling in IaaS clouds.

Unlike all the aforementioned methods, we propose a hybrid algorithm combining genetic algorithm, artificial bee colony optimization and decoding heuristic for scheduling scientific workflows over the available cloud resources.

The proposed algorithm aims to optimize the makespan and economic cost of running scientific workflow applications in an IaaS cloud offering a limited amount of instances and a flexible combination of instance types.

III. PROBLEM MODELING

In this section, we first present the system model. Next, we describe the cloud computing model. Then, we present our workflow model. Finally, we conclude this section by describing the scheduling model formalized as a multi-objective optimization problem.

A. SYSTEM MODEL

Our proposed system model consists of three layers, namely, cloud user, workflow scheduler, and cloud resource, as shown in Figure 1. The first layer is the cloud user, which submit tasks associated with the workflow to the workflow scheduler. The second layer is the workflow scheduler, which contains HGAABC scheduling algorithm. The last layer of the proposed model is the cloud resource in IaaS cloud environment. The working of our proposed system model is explained step by step below.

1. The cloud user submits the workflow to the queue which works on a First-Come-First-Serve (FCFS) basis.
2. The workflows are sent to the proposed HGAABC algorithm, which is used to find the optimal solution for two conflicting objectives, ie. makespan and cost.
3. The schedule generated by the HGAABC algorithm is then dispatched for execution to the cloud environment.
4. After execution, the results are collected and then returned to the corresponding user.

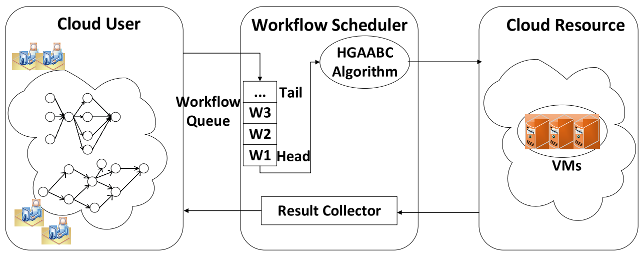


FIGURE 1. System model.

B. IaaS CLOUD MODEL

Suppose that the IaaS provider provides a computation service such as Amazon Elastic Compute Cloud (EC2) [19] which we can use to execute the workflow tasks, and a storage service such as Amazon Elastic Block Store (EBS) [20] which can be attached to the computation resources as a local storage device to offer enough space for the data files resulting from executing tasks. Our hardware platform is represented by a set V of m heterogeneous VMs, which can be of any type as provided by an IaaS provider. Each virtual machine has its own CPU processing speed measured in million floating-point operations per second (MFLOPS), memory in Gigabytes (GB), storage space in GB, bandwidth in Megabits per second (Mbps), computation cost in dollars per hours and storage cost in dollars per minutes per GB.

The users are charged based on the number of time intervals that they have used the resource, even if they have not entirely used the last time interval. For instance, for a 60 seconds interval, if a VM is used for 61 seconds, the user will pay for two periods of 60 seconds, that is, 120 seconds.

All the VMs offered by IaaS providers are assumed to be in the same physical region, so the average bandwidth between VMs, denoted by BW, is roughly the same. In theory a user can access an infinite pool of VMs, but in practice, most IaaS providers restrict the total number of allocated VMs to a user. For example, Amazon currently restricts its common users to a maximum of 20 VMs of EC2 services. Therefore, we assume that a user can acquire a maximum of r VMs simultaneously.

C. WORKFLOW APPLICATION MODEL

A scientific workflow application is modeled as a directed acyclic graph (DAG), defined by a two-tuple $W(T, E)$, where T is the set of vertices standing for n different tasks of the workflow, and E is the set of directed edges between the vertices standing for dependencies. A dependency $e_{i,j}$ between the tasks t_i and t_j represents a precedence constraint that the task t_j cannot start its execution before t_i completes and sends all the needed output data to task t_j. For a given dependency $e_{i,j}$, task t_i is called one of the immediate predecessors of task t_j, and task t_j is called one of the immediate successors of task t_i. A task t_i may have multiple predecessor and multiple successor tasks, defined as $pred(t_i)$ and $succ(t_i)$ of t_i, respectively. A task is viewed as a ready task when all its predecessors have completed execution. Assume that each task t_i has a workload, represented by WL_i, which is the length of the task in machine instructions. Also, each edge $e_{i,j}$ has a weight that indicates the size of the data to be transferred from t_i to t_j, denoted by $DS_{i,j}$.

D. SCHEDULING MODEL

This work focuses on finding a schedule to execute a workflow on IaaS computing resources such that the overall completion time and the overall execution cost are minimized. The overall completion time of a workflow is also called makespan. Let $A = (a_{i,j})_{n\times m}$ be the workflow assignment matrix. If task t_i is assigned to VM v_j, then $a_{ij} = 1$; otherwise, $a_{ij} = 0$. After task t_i is assigned to VM v_j, its execution time $TE\left(t_i, v_j\right)$ can be calculated as

$$TE_{t_i,v_j} = WL_{t_i} \big/ S_{v_j} \tag{1}$$

where S_{v_j} represents the processing speed of VM v_j. The time it takes to transfer data from t_k to t_i can be calculated as

$$DT_{k,i} = \begin{cases} 0 & \textit{if } t_i \textit{ and } t_k \textit{ are assigned to} \\ & \textit{the same VM} \\ DS_{k,i}/BW & \textit{otherwise} \end{cases} \tag{2}$$

Note that the data transfer time between two tasks being executed on the same VM is *0*. Thus, the start time ST_{t_i,v_j} and completion time FT_{t_i,v_j} of task t_i on VM v_j are computed recursively as shown in (3) and (4), respectively.

$$ST_{t_i,v_j} = \begin{cases} A(j) & pred(t_i) = \emptyset \\ max\left\{max_{t_k \in pred(t_i)}\left(FT_{t_k,\bar{v}_k} + DT_{k,i}\right), A(j)\right\} & otherwise \end{cases} \quad (3)$$

$$FT_{t_i,v_j} = TE_{t_i,v_j} + ST_{t_i,v_j} \quad (4)$$

In (3), $A(j)$ denotes the earliest time at which VM v_j is available to begin executing task t_i and $\bar{v}_k$ represents the VM on which task t_k runs. The workflow makespan *MT* is finally defined as the maximum completion time of all the tasks in the workflow. This is depicted in (5).

$$MT = max_{i\in\{1,2,\ldots,n\}} \sum_{j=1}^{m} (a_{i,j} \cdot FT_{t_i,v_j}) \quad (5)$$

The overall execution cost of a workflow consists of the computation cost, the storage cost and the cost of data transfer. Many IaaS providers do not charge for the internal data transfer, so the data transfer cost is assumed to be zero in our cost model. Let the binary variable y_j indicate whether VM j is leased or not. The computation cost *PC* of running a cloud workflow can be defined as follows.

$$PC = \sum_{j=1}^{m} [y_j \cdot \left\lceil \frac{ETL_j - STL_j}{3600} \right\rceil \cdot RP_j] \quad (6)$$

where RP_j is the price per hours of usage for VM v_j, ETL_j and STL_j represent the lease start and lease end times for VM v_j, respectively. The data storage cost *SC* can be given by

$$SC = \sum_{i=1}^{n} \sum_{j=1}^{m} [a_{i,j} \cdot SA_{t_i} \cdot \left\lceil \frac{TE_{t_i,v_j} + \sum_{t_k \in succ(t_i)} DT_{i,k}}{60} \right\rceil \cdot SP_j] \quad (7)$$

where SA_{t_i} is data sizes resulting from executing task t_i, and SP_j denotes the price per GB of data storage per minutes for VM v_j. Finally, the cost of executing the entire workflow, denoted by *TC*, can be given by

$$TC = PC + SC \quad (8)$$

Let T_{v_j} be the set of tasks allocated to VM v_j. Based on the previous definitions, the workflow scheduling problem can be formulated as a mathematical optimization problem, which is described as follows.

$$\textbf{Minimize } MT = max_{i\in\{1,2,\ldots,n\}} \sum_{j=1}^{m} (a_{i,j} \cdot FT_{t_i,v_j}) \quad (9)$$

$$\textbf{Minimize } TC = PC + SC \quad (10)$$

$$\textbf{Subject to } FT_{t_k,\bar{v}_k} + DT_{k,i} \le ST_{t_i,v_j} \quad \forall v_j \in V,\ \forall t_i \in T_{v_j},\ \forall t_k \in pred(t_i) \quad (11)$$

$$\sum_{j=1}^{m} a_{ij} = 1 \quad \forall t_i \in T \quad (12)$$

$$\sum_{j=1}^{m} y_j \le r \quad (13)$$

$$y_j, a_{i,j} \in \{0,1\} \quad \forall t_i \in T,\ \forall v_j \in V \quad (14)$$

Two objective functions given by (9) and (10) aim to find the schedule for a scientific workflow application that minimizes both makespan and cost simultaneously. Constraint (11) ensures that a task can only be started after its predecessor task has finished and all the required input data is received. Constraint (12) ensures that each task is assigned to only one of VMs. Constraint (13) ensures that a maximum of r VMs can be simultaneously acquired. Constraint (14) defines the domain of the variables of the problem.

IV. THE PROPOSED HYBRID ALGORITHM

Because of the NP-Hard nature of the optimization problem described above, there is a great difficulty to find the best solutions in practically acceptable times. This section will show how to apply a hybrid algorithm based genetic algorithm, artificial bee colony optimization, and decoding heuristic to efficiently search for good solutions in large solution spaces. The decoding heuristic generates a feasible scheduling based on a given mapping of tasks to VMs and a mapping of VMs to their types. The hybrid algorithm based on ABC and GA is used to evolve such mappings. The main steps of the HGAABC algorithm is depicted in Algorithm 1. The proposed algorithm begins with generating a random population of *FN* candidate solutions. Each solution is an allocation of the whole workflow tasks over the available VMs. The initial population is passed through the GA algorithm with the first half of the predefined iterations (*MCN*). In the GA algorithm, the solutions are called chromosomes; the chromosomes are recursively enhanced by the GA operators (i.e., selection, crossover, and mutation). The resulting chromosomes are passed to the ABC algorithm at the second half of *MCN*. In the ABC algorithm, the solutions are called food sources; the food sources are enhanced gradually at each iteration through three groups of bees: employed bees, onlooker bees and scout bees. In the evolutionary process, the fast non-dominated sorting method and crowding distance measure are used to update the population. The following subsections describe the phases of the proposed HGAABC algorithm.

A. ENCODING SCHEME

In the proposed algorithm, a solution is a two-tuple including a mapping *TV* of tasks to VMs and a mapping *VK* of VMs to their types. We split a solution into two parts to represent them respectively. The first part *task2vm* is a vector representing the mapping *TV* and its length is equal to the number n of tasks in the workflow. The second part *vm2type* is a vector denoting the mapping *VK* and its length is equal to the maximum number r of available VMs. The value of each element in the

Algorithm 1 The Proposed HGAABC Algorithm

```
Input: Workflow, set of VMs and set of VM types
Output: Set of Pareto-optimal solutions
/* Initialization phase*/
1. Randomly generate an initial population of FN individuals. (see Section
IV Part C for details)
/*GA phase*/(see Section IV Part D for details)
2. Repeat
3.    Choose FN/2 pairs of chromosomes from the current
       population by using selection operator.
4.    Apply the crossover operator to each of the selected
      pairs in Step 3 to generate FN chromosomes with a
      predefined crossover probability CP.
5.    Apply the mutation operator to each of the generated
      FN chromosomes with a predefined mutation
      probability MP.
6.   Create a temporary population by combining the parent
      population and the offspring population generated by
       the GA operators.
7.    Create a new population P1 by choosing FN
      chromosomes from the temporary population based
      on the elitist preservation strategy. (see Section IV
      Part F for details)
8. Until the number of cycles reaches to the half number of
    predefined iterations
/*ABC phase*/ (see Section IV Part E for details)
9. Repeat
10.   Send the employed bees onto their food sources
       according to the elite-guide strategy.
11.   Create a temporary population by combining the
       current population and the population generated
       by the employed bees
12.   Create a new population P2 by choosing FN food
       sources from the temporary population based
       on the elitist preservation strategy. (see Section IV
       Part F for details)
13.   Send the onlooker bees onto their food sources
      depending on their nectar amounts and the elite-guide
       strategy.
14.   Create a temporary population by combining the
       population P2 and the population generated by the
       onlooker bees.
15.   Create a new population P3 by choosing FN food
       sources from the temporary population based on the
       elitist preservation strategy. (see Section IV Part F
       for details)
16.   Send the scout bees to search possible new food
       sources.
17. Until the number of cycles reaches to the half number
    of predefined iterations
18. Return all non-dominated solutions in the current
    population
```

vector *task2vm* is a real number between *0* and the number of VMs available to run the tasks. The value of each element in the vector *vm2type* is a real number between *0* and the number of VM types available. During the search process, the solutions in continuous representation are used to find the non-dominated solutions. when decoding the solution, we apply the floor function to convert the continuous vectors into the discrete vectors. In the discrete vector *task2vm,* each index denotes a task and its value denotes the VM where this task will be executed. In the discrete vector *vm2type,* each index represents a VM and its value represents the VM type. For example, Fig. 2 shows the encoding of a schedule with 5 VMs, 6 tasks and 4 types of VMs.

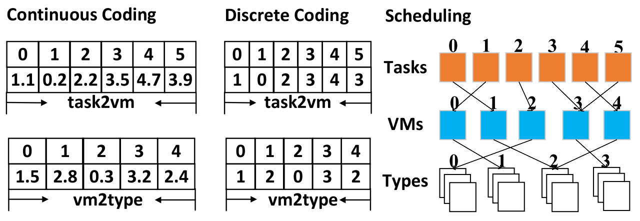


FIGURE 2. Encoding scheme of a valid schedule.

B. DECODING SCHEME

Decoding a solution means to generate a feasible schedule. We define a schedule $S = (R, M, MT, TC)$ in terms of a set of VMs actually leased, a task to VM mapping, the overall completion time and the overall cost. For a given solution in discrete representation, we apply the decoding heuristic in Algorithm 2 to decode the solution into an actual schedule. Initially, the set of VMs actually leased R and the set of task to VM mappings M are empty and the overall makespan MT and the overall cost TC are set to zero. The initial lease start time STL_j and lease end time ETL_j of all VMs are set to zero. After this, the algorithm estimates the start and end time of each workflow task. The next step is the calculation of the total data transfer time ($Transtime_{t_i}$), data storage size ($SA_{t_i,}$), and cost (SC). Then the lease start and end time can be estimated for each leased VM. Once the algorithm finishes processing each task, R will contain all VMs that need to be leased as well as the times when they should be started and shutdown. Additionally, the entire task to VM mapping will be in M and each task will have a VM assigned to it as well as an estimated start and end times. With this information, the algorithm can now use (5) and (8) to compute the overall cost TC and completion time MT associated to the current solution. Finally, the algorithm returns the schedule associated to the given solution.

C. INITIALIZING POPULATION

In the proposed algorithm, the initial population is generated by using a random strategy. Let $X_i = \{x_{i1}, x_{i2}, \cdots, x_{iq}\}$ denotes the ith individual in the population, where q is the

Algorithm 2 The Decoding Heuristic

```
Input: An array $L[n]$ denoting the mapping $task2vm$
    An array $Q[r]$ denoting the mapping $vm2type$
Output: A schedule $S = (R, M, MT, TC)$
1.  $R = \phi$
2.  $M = \phi$
3.  $MT = 0$
4.  $TC = 0$
5.  $SC = 0$
6.  For $j = 0$ to $r - 1$
7.    $STL_j = 0$
8.    $ETL_j = 0$
9.  EndFor
10. For $i = 0$ to $n - 1$
11.   $Transtime_{t_i} = 0$
12.   $SA_{t_i} = 0$
13.   If $t_i$ has no predecessor tasks
14.     $ST_{t_i,v_{L[i]}} = ETL_{v_{L[i]}}$
15.   Else
16.     $ST_{t_i,v_{L[i]}} = \max\left\{\max_{t_k \in pred(t_i)}\left(FT_{t_k,\bar{v}_k} + DT_{k,i}\right), ETL_{v_{L[i]}}\right\}$
17.   EndIf
18.   Calculate $TE_{t_i,v_{L[i]}}$ according to (1)
19.   Calculate $FT_{t_i,v_{L[i]}}$ according to (4)
20.   For each successor task $t_s$ of $t_i$
21.     If $t_s$ is allocated to a VM different to $v_{L[i]}$
22.       $Transtime_{t_i} = Transtime_{t_i} + DT_{i,s}$
23.     EndIf
24.     $SA_{t_i} = SA_{t_i} + DS_{i,s}$
25.   EndFor
26.   For each predecessor task $t_k$ of $t_i$
27.     $SA_{t_i} = SA_{t_i} + DS_{k,i}$
28.   EndFor
29.   $SC = SC + SA_{t_i} \cdot \left\lceil \frac{TE_{t_i,v_{L[i]}} + Transtime_{t_i}}{60} \right\rceil \cdot SP_{Q[L[i]]}$
30.   Add the mapping of task $t_i$ to VM $v_{L[i]}$ to $M$
31.   If $v_{L[i]} \notin R$
32.     $STL_{v_{L[i]}} = ST_{t_i}$
33.     Add VM $v_{L[i]}$ to $R$
34.   EndIf
35.   $ETL_{v_{L[i]}} = ST_{t_i} + ET_{t_i} + Transtime_{t_i}$
36. Endfor
37. Calculate $MT$ according to (5)
38. Calculate $TC$ according to (8)
39. Return $S$
```

problem dimension and its size is equal to $n + r$. Each individual is initialized through the following equation.

$$x_{ij} = L_j + rand[0, 1](U_j - L_j) \tag{15}$$

where $rand[0, 1]$ is a random number chosen from a normal distribution within the range from zero and one, and L_j and U_j are the lower and upper bounds for the dimension j, respectively.

D. APPLYING THE GA ALGORITHM

At the first phase, the GA is applied to find the non-dominated solutions to the workflow scheduling problem. In each iteration, the GA delivers the chromosomes between three different operators: the selection, crossover, and mutation operators. These operators are explained in the following subsections respectively.

1) SELECTION OPERATOR

In the GA algorithm, not all the chromosomes in the current population are evolved through the GA operators in each iteration. The roulette wheel strategy [21] is chosen as the selection mechanism. Each chromosome in the current population has an associated selection probability $GP(X_i)$ which is define as

$$GP(X_i) = f(X_i) \Big/ \sum\nolimits_{i=1}^{FN} f(X_i) \tag{16}$$

where $f(X_i)$ represents the fitness value of chromosome X_i and it can be calculated using the following equation.

$$f(X_i) = (FN - R(X_i) + 1) \Big/ \sum\nolimits_{j=1}^{FN} [FN - R(X_j) + 1] \tag{17}$$

where $R(X_i)$ denotes the rank of a chromosome X_i in the population and it can be given by

$$R(X_i) = 1 + d_i \tag{18}$$

where d_i denotes the number of chromosomes dominating the chromosome X_i in the current population. It is worth remarking that chromosomes with an identical rank obtain the same fitness value and have thus the same probability to reproduce.

2) CROSSOVER OPERATOR

The crossover operator aims to produce new chromosomes by changing the position of the genes inside every two chromosomes. In the crossover, the two points randomly selected as showed in Fig. 3, divide the parent into three parts (0–1, 1–2, 2–3). All genes from the parts (0–1, 2–3) are copied to identical positions in the offspring and all the genes within the part (1–2) of two parent are exchanged for generating the remaining parts of two offspring.

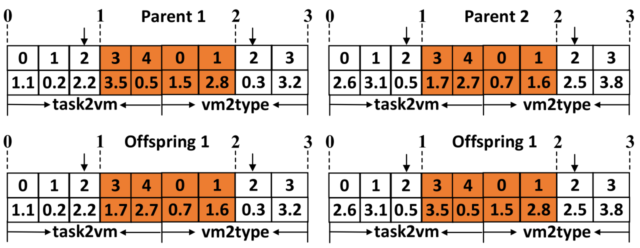


FIGURE 3. Crossover operator.

3) MUTATION OPERATOR

The mutation operator is used to generate small perturbations on chromosomes in order to maintain the diversity of population. Uniform mutation operator is used in this paper.

In uniform mutation, a random selected gene is replaced with a random real value between its lower and upper bounds. Fig. 4 shows how the mutation operator operates.

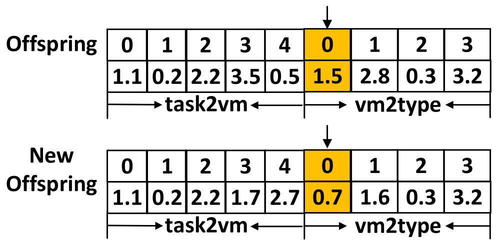


FIGURE 4. Mutation operator.

E. APPLYING THE ABC ALGORITHM

The solutions that are returned from the GA algorithm are fed into the ABC algorithm to find the non-dominated solutions from the GA generated solutions. In the ABC algorithm, the solutions are called food sources. The number of the employed bees or the onlooker bees is equal to the number of food sources in the population. The ABC algorithm consists of three stages as follows.

1) SENDING EMPLOYED BEES

The employed bees generate food sources in the neighborhood of their current locations. In the original ABC algorithm, the new candidate food source is generated by moving the old food source towards another food source selected randomly from the population. Since the probability that the randomly chosen solution is a good solution is the same as that the randomly chosen solution is a bad one, the new candidate solution is not promising to be a solution better than the previous one. In order to improve the exploitation, we take advantage of the information of the non-dominated solution to guide the search of candidate solutions. The local search technique is called the elite-guide strategy. Each employed bee X_i generates a new candidate solution V_i in the neighborhood of its present position as follows:

$$V_{ij}=X_{ij}+\varphi_{ij}\left(X_{ij}-X_{kj}\right)+\omega_{ij}\left(Y_j-X_{ij}\right)\ Y\in ES \tag{19}$$

where X_k is a randomly selected food source which is different from i, j is a random dimension index chosen from the set$\{1,2,\ldots,q\}$, and φ_{ij} is a uniformly distributed real random number within the range $[-1, 1]$. The third term in the right-hand side of (19) is called the elite-guide term, where Y_j denotes the jth element of the solution choosing from ES randomly. In (19), ω_{ij} is a random number in the range of [0, 1.5], and ES denotes the elite set which consist of the solutions in the first non-dominated front of the current population. When the value obtained by (19) exceed its predetermined boundaries, it is set to its boundaries. After the candidate solution is generated, its fitness is assessed. If the fitness of the candidate solution is better than that of the old one, then candidate solution replaces the old one, and the trial for this candidate solution is set to zero. Otherwise, the trial for the old solution is incremented by one. After all the employed bees independently perform local search, the parent population and offspring population are combined into a population. This combined population undergoes a sorting based on the elitist preservation strategy to determine the next population.

2) SENDING ONLOOKER BEES

After all the employed bees complete their search processes, they come into the hive and share the nectar information of the food sources and their position information with the onlooker bees on the dance area. Each onlooker bee selects a food source depending on the probability value associated with the food source. The selecting probability $AP(X_i)$ is calculated as follows [22].

$$AP\left(X_i\right)=0.9\times f\left(X_i\right)\Big/\max_{k\in\{1,2,..,FN\}} f\left(X_k\right)+0.1 \tag{20}$$

And then a random real number within the range [0, 1] is generated for each food source. If the number is greater than the probability value associated with that source then the onlooker bee keeps the old positions, otherwise the onlooker bee updates its position as the employed bee does. After all onlookers complete the exploitation process, there are 2^*FN solutions. The population selection strategy also will be applied to choose FN optimal solutions as the new population.

3) SENDING SCOUT BEES

A scout bee performs randomly search to increase the population diversity and avoid being trapped in local optimum. If a food source cannot be improved further through a predetermined number LIM of trials, then that food source will be abandoned. The employed bee corresponding to the abandoned food source becomes a scout bee, and the food source is replaced with a solution that is produced in the same manner as that in the initialization phase.

F. ELITIST PRESERVATION STRATEGY

In order to preserve the better solutions generated during the evolution process, the parent and offspring population after evolution are combined into a population. The elitist preservation strategy based on fast non-dominated sorting and crowding distance measure proposed by Deb *et al.* [23] is applied to select FN feasible solutions from the combined population in order to generate a new population.

In the proposed elitist preservation strategy, the first step is to choose the members of the new population from the non-dominated fronts in the order of their ranking. Assume that the front f_l denotes the last non-dominated front beyond which no other fronts can be accommodated in the new population. If the number of solutions in all fronts from the first front f_1 to the last front f_l is greater than the population size, the crowding distance measure is used to remove the excess individuals from the last front f_l.

V. PERFORMANCE EVALUATION

A. EXPERIMENT SETUP

To evaluate the performance of the proposed HGAABC algorithm in addressing the problem of scientific workflow scheduling in IaaS clouds, we use an extension of CloudSim [24] called the WorkflowSim-1.0 toolkit [25] to simulate a cloud environment. Pegasus project has published five real-world workflow applications from diverse scientific areas, four of which are used in our experiments. They are Montage workflow for astronomical physics, Sipht workflow for bioinformatics, Cybershake workflow for earthquake hazards and LIGO workflows for detecting gravitational waves. Fig. 5 describes the simplified structures of small instances of workflows used in our experiments and the detailed characteristics of these workflows can be found in [26].

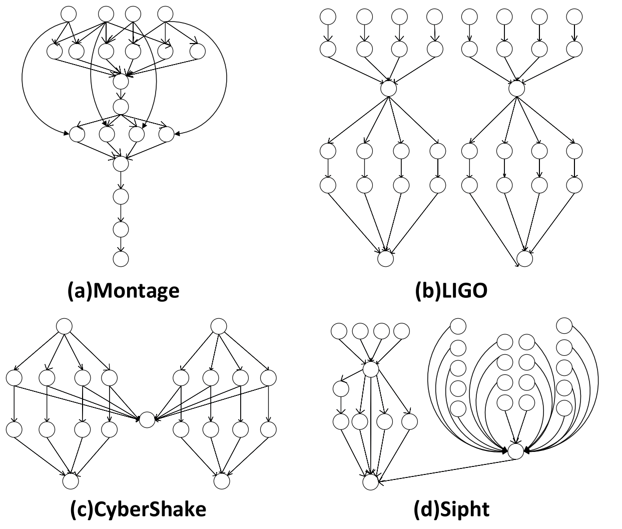


FIGURE 5. Structure of the four different workflows.

We modeled an IaaS provider providing a single data center and five different types of VMs. The configurations of each type of VM are based on Amazon EC2 offerings and are shown in Table 1. The maximum number of VMs that can be simultaneously rented is limited to 20. We used the method proposed by Ostermann *et al.* [27] to estimate the computing capacity in MFLOPS based on the number of EC2 compute units. The price for data storage is 0.10 \$ per GB-month. The workload of each task is randomly generated from 10000 to 100000 MFLOPS and the output size is randomly generated from the interval of [10, 100]GB. The average bandwidth between VMs is set to 0.1 GB/s.

TABLE 1. Type of VMs used in the experiments.

Type	EC2 Units	Computing Capacity (MFLOPS)	Cost per Hour
M1.small	1	4400	\$0.06
M1.medium	2	8800	\$0.12
M1.large	4	17600	\$0.24
M1.xlarge	8	35200	\$0.48
M3.xlarge	13	57200	\$0.50

The performance of the proposed HGAABC algorithm is compared to that of a multi-objective ant colony optimization (PBACO) algorithm proposed in [5], a hybrid multi-objective particle swarm optimization (HPSO) algorithm proposed in [4] and a single-objective HEFT algorithm proposed in [7]. In addition, we also compare the HGAABC algorithm with its variants, including a multi-objective genetic algorithm (MGA) and a multi-objective artificial bee colony algorithm (MABC). The programs for all six algorithms were coded in the Java language and ran on an Intel(R) Core(TM) i7 processor with 1.80 GHz CPU and 8 GB RAM. The settings for various parameters in the HGAABC algorithm have a direct effect on the algorithm performance. Appropriate parameter values are determined on the basis of preliminary experiments. The final parameter settings are determined to be $FN = 50$, $MCN = 1000$, $LIM = 200$, $CP = 0.8$ and $MP = 0.5$. In the case of the HPSO algorithm, the corresponding parameters are population size = 50, maximum iterations = 1000, $c1 = 2.5 \rightarrow 0.5$ and $c2 = 0.5 \rightarrow 2.5$, inertia weight $\omega = 0.9 \rightarrow 0.1$. The parametric values for the PBACO algorithm are set to be population size = 50, maximum iterations = 1000, $\alpha = 3$, $\beta = 2$, $\rho = 0.01$. The parametric values for the MGA algorithm are set to be population size = 50, maximum iterations = 1000, $CP = 0.8$, and $MP = 0.5$. The parametric values for the MGA algorithm are set to be population size = 50, maximum iterations = 1000, and $LIM = 200$. For all the experiments, 20 independent runs were conducted to guarantee statistical validity of the results.

B. SIMULATION RESULTS

In the first series of experiments, we assess the behavior of the different algorithms optimizing each individual objection. Because of the lack of information about the preferences of objectives, we utilize a Fuzzy-based approach [28] to generate the best compromised solution from the obtained Pareto set. For each objective function f_k, a linear membership function μ_k is defined as follows:

$$\mu_k = \begin{cases} 1 & f_k \le f_k^{min} \\ (f_k^{max} - f_k)/(f_k^{max} - f_k^{min}) & f_k^{min} < f_k < f_k^{max} \\ 0 & f_k \ge f_k^{max} \end{cases} \tag{21}$$

where f_k^{min} and f_k^{max} represent the maximum and minimum values of the kth objective function among all non-dominated solutions, respectively. Correspondingly, the normalized membership function μ^i is calculated for each non-dominated solution as

$$\mu^i = \sum_{k=1}^{OB} \mu_k^i / \sum_{j=1}^{ND} \sum_{k=1}^{OB} \mu_k^i \tag{22}$$

where OB and ND represent the number of objectives functions and non-dominated solutions, respectively. The solution with the maximum membership μ^i is chosen as the best compromise solution. Fig. 6 and Fig. 7 show the average

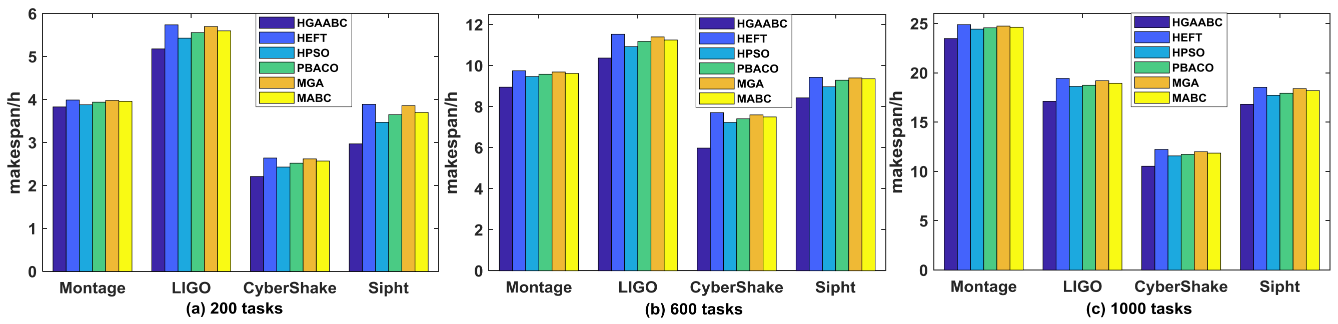


FIGURE 6. Comparison of average makespan for different algorithms.

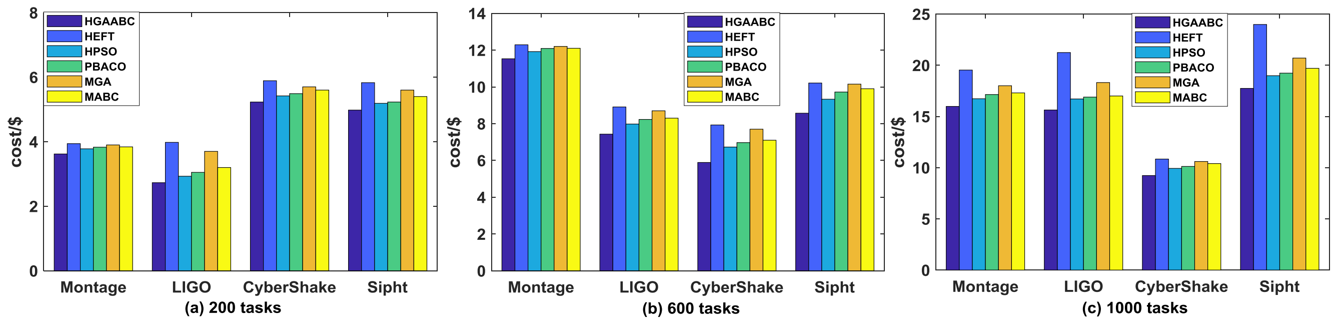


FIGURE 7. Comparison of average cost for different algorithms.

makespan and cost of the schedules computed by the six algorithms for four types of workflows with varying the number of tasks, respectively. From Figs. 6 and 7, it can be seen that HEFT algorithm performs worst in terms of the overall makespan and cost for four types of workflows. Furthermore, it can also be observed that HGAABC algorithm outperforms PBACO, HPSO, MGA, MABC and HEFT algorithms in terms of the overall makespan and cost for four types of workflows. This is because HGAABC can search the solution space more efficiently and globally so that it can obtain both relatively low cost and relatively short makespan.

In the second series of experiments, we analyze the tradeoff solutions computed by HPSO, PBACO, MGA, MABC, and HGAABC for different workflow types and apply Q-metric [29], FS-metric [30] and S-metric [31] to compare the quality of the Pareto-optimal fronts among different algorithms. Q-metric is utilized to measure the convergence of the non-dominated solutions found by the two algorithms, which is computed by

$$Q(A,B) = |\varphi| \big/ |\gamma| \tag{23}$$

where A and B represent two sets of Pareto-optimal solutions found by two different algorithms, respectively. γ is the set of non-dominated solutions within A∪B and $\varphi = A \cap \gamma$. The Pareto-optimal front found by an algorithm has better convergence to the true Pareto-optimal front than that found by the other algorithm, if and only if $Q(A,B) > 0.5$. FS-metric indicates the size of the space covered by the Pareto-optimal front found by an algorithm, which is computed by

$$FS(A) = \sqrt{\sum_{i=1}^{m} \min_{(x_1,x_2)\in A\times A} (f_i(x_1) - f_i(x_2))^2} \tag{24}$$

where f is the objection function and m is the number of objectives. A larger FS-metric value is preferable, which means that the Pareto-optimal solutions found by an algorithm are widely spread along the true Pareto front. S-metric is employed to evaluate the uniformity of the Pareto-optimal solutions found by an algorithm, which is computed by

$$S(A) = \sqrt{\sum_{i=1}^{|A|} (d_i - \bar{d})^2 \big/ |A|} \tag{25}$$

where $d_i = \min_{k\in A \wedge k\neq i} \sum_{j=1}^{m} |f_j(k) - f_j(i)|$ and $\bar{d} = \sum_{i=1}^{|A|} d_i \big/ |A|$. The desired value for S-metric is zero, which means that the Pareto-optimal solutions found by an algorithm are equidistantly spaced.

Fig. 8 shows the tradeoff solutions obtained by five algorithms for four types of workflow applications with 1000 tasks. The figure shows that HGAABC algorithm has the optimal trade-off solutions, and MGA algorithm performs the worst. Tables 2-5 show the comparison of results among five algorithms considering Q-metric, FS-metric and S-metric for Montage, CyberShake, Sipht, and LIGO workflows, respectively. The "true" value in the first five rows means that one algorithm is better than another algorithm on Q-metric. As can be seen from the table, for the Q-metric, the Pareto-optimal solutions found by HGAABC algorithm

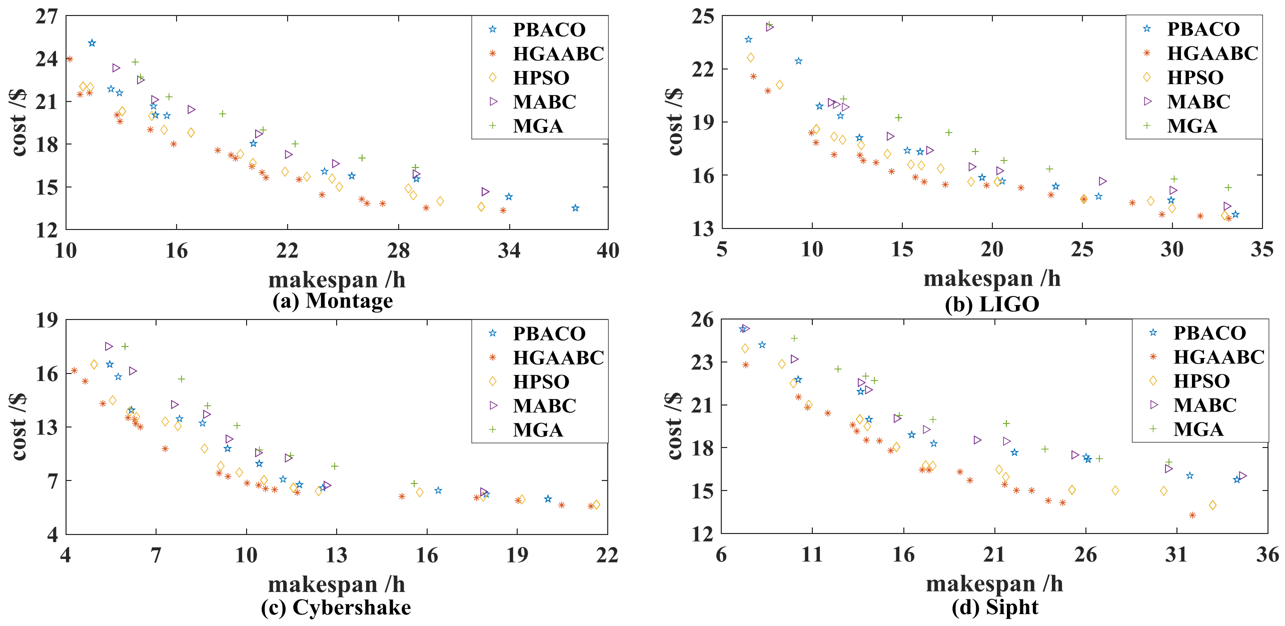


FIGURE 8. Makespan-cost trade-offs on four types of scientific workflows.

TABLE 2. Multi-objective performance metrics on the montage workflow.

Q-metric	HGAABC	HPSO	PBACO	MABC	MGA
HGAABC	——	True	True	True	True
HPSO	——	——	True	True	True
PBACO	——	——	——	True	True
MABC	——	——	——	——	True
FS-metric	0.14	0.09	0.05	0.03	0.02
S-metric	0.53	0.74	1.16	1.31	1.35

TABLE 3. Multi-objective performance metrics on the LIGO workflow.

Q-metric	HGAABC	HPSO	PBACO	MABC	MGA
HGAABC	——	True	True	True	True
HPSO	——	——	True	True	True
PBACO	——	——	——	True	True
MABC	——	——	——	——	True
FS-metric	0.39	0.14	0.09	0.07	0.04
S-metric	0.28	0.52	1.09	1.26	1.34

TABLE 4. Multi-objective performance metrics on the CyberShake workflow.

Q-metric	HGAABC	HPSO	PBACO	MABC	MGA
HGAABC	——	True	True	True	True
HPSO	——	——	True	True	True
PBACO	——	——	——	True	True
MABC	——	——	——	——	True
FS-metric	0.54	0.37	0.16	0.11	0.05
S-metric	0.39	0.84	1.38	1.64	1.72

TABLE 5. Multi-objective performance metrics on the sipht workflow.

Q-metric	HGAABC	HPSO	PBACO	MABC	MGA
HGAABC	——	True	True	True	True
HPSO	——	——	True	True	True
PBACO	——	——	——	True	True
MABC	——	——	——	——	True
FS-metric	0.15	0.11	0.07	0.04	0.03
S-metric	0.96	1.28	2.22	2.47	2.61

always outperform that of four other algorithms, which indicates that the HGAABC algorithm has better convergence than the comparative algorithms. Regarding FS-metric, the value of HGAABC algorithm is greater than the corresponding values for four other algorithms. It means that the Pareto-optimal solutions found by the HGAABC algorithm have better diversity than that found by the comparative algorithms. With respect to the S-metric, the value of HGAABC algorithm is less than the values of four other algorithms. It shows that HGAABC algorithm achieves the better uniformity of Pareto-optimal front than the other four algorithms.

Due to the stochastic characteristic of the HGAABC, HPSO, PBACO, MGA, and MABC algorithms, the statistical test should be conducted in order to validate the statistical significance of the achieved experimental results. In the third series of experiments, we use wilcoxon signed rank test [32] to find out whether there is any significant difference between the results obtained by five multi-objective algorithms. This test includes a null hypothesis H_0 and an alternate hypothesis H_1 defined as

$$H_0 : \mu_1 = \mu_2 \tag{26}$$

$$H_1 : \mu_1 \neq \mu_2 \tag{27}$$

where H_0 is a statement of no significant difference between two algorithms and H_1 denotes the presence of a significant difference between two algorithms. During the test, a level of significance α is set to 0.05. If the$p$-value is less than (or equal to) 0.05, then H_0 is rejected in favor of H_1. But,

TABLE 6. Results of Wilcoxon signed rank test for FS-metric.

Workflow	Number of tasks	HGAABC vs. HPSO			HGAABC vs. PBACO			HGAABC vs. MABC			HGAABC vs. MGA		
		R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win
Montage	200	166	5	4.55e-04/>	206	4	1.62e-04/>	143	28	1.12e-02/>	207	3	1.61e-04/>
	400	210	0	8.8e-05/>	183	7	3.96e-04/>	163	8	7.73e-04/>	163	27	3.76e-04/>
	600	161	10	1.06e-03/>	178	12	8.33e-04/>	161	10	1.06e-03/>	210	0	8.8e-05/>
	800	210	0	8.8e-05/>	201	9	3.36e-04/>	166	5	4.47e-04/>	200	10	3.34e-04/>
	1000	153	18	3.28-04/>	182	8	4.62e-04/>	187	10	1.64e-04/>	192	18	4.42e-04/>
LIGO	200	210	0	8.8e-05/>	163	8	7.73e-04/>	183	15	1.21e-03/>	142	11	1.92e-03/>
	400	159	12	1.36e-03/>	171	0	8.8e-05/>	133	38	3.85e-02/>	171	30	6.35e-05/>
	600	210	0	8.8e-05/>	161	10	1.06e-03/>	202	8	3.25e-04/>	181	20	1.53e-03/>
	800	142	11	1.92e-03/>	210	0	8.8e-05/>	210	0	8.8e-05/>	210	0	8.8e-05/>
	1000	210	0	8.8e-05/>	142	11	1.92e-03/>	210	0	8.8e-05/>	210	0	8.8e-05/>
CyberShake	200	187	3	2.21e-04/>	188	2	1.81e-04/>	166	5	4.55e-04/>	133	38	3.85e-02/>
	400	165	66	3.95e-01/=	210	0	8.8e-05/>	185	25	3.95e-04/>	169	31	2.87e-03/>
	600	210	0	8.8e-05/>	160.5	10.5	1.08e-03/>	203	7	2.53e-04/>	161	11	1.18e-03/>
	800	112	24	2.28e-02/>	168	3	3.24e-04/>	118	12	2.31e-02/>	158	43	2.24e-03/>
	1000	125	65	2.27e-01/=	114	22	1.72e-02/>	210	0	8.8e-05/>	154	22	2.32e-02/>
Sipht	200	210	0	8.8e-05/>	210	0	8.8e-05/>	140	65	2.21e-01/=	105	31	5.55e-02/>
	400	155	16	2.46e-03/>	195.5	14.5	7.2e-04/>	210	0	8.8e-05/>	210	0	8.8e-05/>
	600	160	11	1.17e-03/>	198	12	5.1e-04/>	210	0	8.8e-05/>	200	10	3.34e-04/>
	800	164	7	6.27e-04/>	201	9	3.36e-04/>	154	17	5.87e-04/>	198	11	3.26e-04/>
	1000	162	9	8.62e-04/>	203	7	2.53e-04/>	170	30	6.62e-04/>	178	12	8.33e-04/>

TABLE 7. Results of Wilcoxon signed rank test for S-metric.

Workflow	Number of tasks	HGAABC vs. HPSO			HGAABC vs. PBACO			HGAABC vs. MABC			HGAABC vs. MGA		
		R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win
Montage	200	210	0	8.8e-05/>	209	1	1.55e-04/>	159	12	1.36e-03/>	210	0	8.8e-05/>
	400	210	0	8.8e-05/>	210	0	8.8e-05/>	187	13	7.42e-05/>	210	0	8.8e-05/>
	600	189	1	1.55e-04/>	141	30	1.56e-02/>	195.5	14.5	7.2e-04/>	141	30	1.56e-02/>
	800	210	0	8.8e-05/>	155	16	2.46e-03/>	183	15	1.21e-03/>	156	21	2.32e-03/>
	1000	191	19	3.77e-03/>	108	63	3.26e-01/>	132	68	1.08e-01/=	108	83	2.26e-01/=
LIGO	200	210	0	8.8e-05/>	188	2	1.82e-04/>	200	10	3.34e-04/>	188	22	1.82e-04/>
	400	197	13	2.58e-04/>	210	0	8.8e-05/>	210	0	8.8e-05/>	167	43	3.82e-03/>
	600	148	23	6.47e-03/>	133	38	3.85e-02/>	150	21	6.38e-03/>	133	38	3.85e-02/>
	800	210	0	8.8e-05/>	210	0	8.8e-05/>	210	0	8.8e-05/>	210	0	8.8e-05/>
	1000	78	93	7.44e-01/<	126	45	7.78e-02/>	210	0	8.8e-05/>	210	0	8.8e-05/>
CyberShake	200	199	11	1.55e-04/>	196	14	1.84e-03/>	187	23	1.97e-04/>	196	14	1.84e-03/>
	400	154.5	16.5	2.65e-03/>	210	0	8.8e-05/>	154.5	16.5	2.65e-03/>	210	0	8.8e-05/>
	600	171	39	4.28e-02/>	210	0	8.8e-05/>	165	45	3.28e-02/>	210	0	8.8e-05/>
	800	210	0	8.8e-05/>	142	29	1.38e-02/>	130	69	1.18e-02/>	142	29	1.38e-02/>
	1000	143	28	1.12e-02/>	178	32	1.12e-02/>	140	48	2.21e-02/>	197	13	1.62e-02/>
Sipht	200	170	1	2.33e-04/>	179	11	7.23e-04/>	173	4	2.43e-04/>	142	11	1.92e-03/>
	400	162	9	8.61e-04/>	192	18	1.18e-03/>	159	11	8.21e-04/>	210	0	8.8e-05/>
	600	210	0	8.8e-05/>	188	2	1.82e-04/>	210	0	8.8e-05/>	188	2	1.82e-04/>
	800	105	31	5.55e-02/>	180	10	6.24e-04/>	210	0	8.8e-05/>	150	44	3.48e-04/>
	1000	180	30	1.56e-02/>	169	21	2.89e-03/>	198	12	5.1e-04/>	210	0	8.8e-05/>

if the *p*-value is greater than 0.05, then H_0 is not rejected. Tables 6, 7 and 8 show the test results for four types of workflow with varying the number of tasks. The *greater-than* sign ($>$) and *the less-than* sign ($<$) denote that HGAABC algorithm performs significantly better and worse than its comparative one, respectively. The equal sign ($=$) represents that there is no significant difference between HGAABC algorithm and its comparative one. R^+ is the sum of ranks for the problems in which HGAABC algorithm outperformed its comparative one, and R^- is the sum of ranks for the opposite. From these tables, it can be obviously seen that HGAABC algorithm has higher "$>$" counts than other comparative algorithms. This implies that HGAABC algorithm shows a significant improvement over the PBACO, HPSO, MABC, and MGA algorithms in terms of Q-metric, FS-metric and S-metric with $\alpha = 0.05$.

In the last series of experiments, we evaluate the scalability of the proposed algorithm. Table 9 shows the execution time of the HGAABC, MGA, and MABC algorithms for four types of workflow applications with different number of tasks. From the table we can see that the execution time of the proposed algorithm is greater than that of the MABC algorithm but smaller than that of the MGA algorithm. The reason is that the predefined number of iterations is divided

TABLE 8. Results of Wilcoxon signed rank test for Q-metric.

Workflow	Number of tasks	HGAABC vs. HPSO			HGAABC vs. PBACO			HGAABC vs. MABC			HGAABC vs. MGA		
		R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win	R^+	R^-	*p*-value/win
Montage	200	161	11	1.08e-03/>	204	6	2.18e-04/>	126	45	7.78e-02/>	198	12	1.78e-04/>
	400	178.5	11.5	7.74e-04/>	163	8	7.33e-04/>	196	14	1.84e-03/>	169	21	2.89e-03/>
	600	166	5	4.47e-04/>	210	0	8.8e-05/>	170	10	4.97e-04/>	195.5	14.5	7.2e-04/>
	800	163	9	7.93e-04/>	210	0	8.8e-05/>	133	38	3.85e-02/>	208	2	8.43e-05/>
	1000	163	8	7.33e-04/>	189	1	1.54e-04/>	174	11	6.97e-04/>	179	31	1.04e-04/>
LIGO	200	210	0	8.8e-05/>	210	0	8.8e-05/>	188	2	1.82e-04/>	210	0	8.8e-05/>
	400	179	10	6.69e-04/>	160	11	1.16e-03/>	143	28	1.12e-02/>	210	0	8.8e-05/>
	600	179.5	10.5	6.69e-04/>	209	1	1.02e-04/>	210	0	8.8e-05/>	112	24	2.28e-02/>
	800	165	6	5.31e-04/>	182	8	4.62e-04/>	210	0	8.8e-05/>	108	63	3.26e-01/>
	1000	210	0	8.8e-05/>	210	0	8.8e-05/>	162	9	8.61e-04/>	210	0	8.8e-05/>
CyberShake	200	163	8	7.35e-04/>	210	0	8.8e-05/>	174	15	7.84e-04/>	153	18	3.28-04/>
	400	162	10	9.27e-04/>	210	0	8.8e-05/>	209	1	1.02e-04/>	210	0	8.8e-05/>
	600	210	0	8.8e-05/>	210	0	8.8e-05/>	187	23	1.97e-04/>	210	0	8.8e-05/>
	800	167	4	3.79e-04/>	163	8	7.33e-04/>	210	0	8.8e-05/>	163	38	6.31e-04/>
	1000	210	0	8.8e-05/>	210	0	8.8e-05/>	126	45	7.78e-02/>	140	60	2.53e-01/>
Sipht	200	210	0	8.8e-05/>	201	9	3.38e-04/>	210	0	8.8e-05/>	205	5	3.28e-04/>
	400	197.5	12.5	5.5e-04/>	207	4	1.5e-04/>	210	0	8.8e-05/>	134	56	2.57e-02/>
	600	164	7	6.25e-04/>	208	2	1.29e-05/>	159	27	5.75e-04/>	182	8	4.62e-04/>
	800	173	18	1.81e-03/>	189	1	1.54e-04/>	184	8	1.74e-03/>	172	18	1.14e-04/>
	1000	187	3	2.1e-04/>	164	7	6.25e-04/>	126	45	7.78e-02/>	210	0	8.8e-05/>

TABLE 9. Running time of scheduling algorithms.

Workflow	Number of tasks	Running time /min		
		HGAABC	MGA	MABC
Montage	200	0.695	1.017	0.534
	400	1.654	1.658	1.223
	600	1.735	2.202	1.434
	800	2.631	3.101	2.598
	1000	3.426	4.753	3.125
LIGO	200	1.021	1.595	0.628
	400	1.743	2.112	1.562
	600	2.924	3.996	2.421
	800	3.948	4.742	2.983
	1000	4.269	5.827	4.091
CyberShake	200	0.654	0.991	0.447
	400	1.125	1.338	1.104
	600	1.775	2.872	1.623
	800	2.879	3.219	2.552
	1000	3.929	4.231	3.065
Sipht	200	0.735	1.025	0.683
	400	1.694	1.936	1.521
	600	2.002	2.372	1.963
	800	2.825	3.037	2.652
	1000	3.483	4.543	3.201

equally between the GA and ABC algorithms. In addition, we also observe that the proposed algorithm takes less than 5 min to solve the difficult workflow scheduling problem of up to 1000 tasks. Therefore, our algorithm can be suitable for large-scale scientific workflow applications.

VI. CONCLUSION

This paper investigates the problem of scientific workflow scheduling in IaaS clouds. The scenario is modeled as a multi-objective optimization problem which aims to optimize the workflow makespan and cost simultaneously and is solved using the hybrid multi-objective optimization algorithm, HGAABC. The proposed algorithm considers fundamental features of IaaS providers such as a pay-as-you-go model, heterogeneity, elasticity, and dynamicity of the resources. The simulation experiments conducted with four well-known scientific workflows show that our algorithm has an overall better performance than the state-of-the-art algorithms.

REFERENCES

[1] A. Rehman, S. S. Hussain, Z. U. Rehman, S. Zia, and S. Shamshirband, "Multi-objective approach of energy efficient workflow scheduling in cloud environments," *Concurrency Comput. Pract. Exper.*, vol. 31, no. 8, p. e4949 Apr. 2019. doi: 10.1002/cpe.4949.

[2] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr./Jun. 2014. doi: 10.1109/TCC.2014.2314655.

[3] Y.-C. Liang, A. H.-L. Chen, and Y.-H. Nien, "Artificial bee colony for workflow scheduling," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Beijing, China, Jul. 2014, pp. 558–564.

[4] 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. doi: 10.1016/j.parco.2017.01.002.

[5] 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. doi: 10.1109/ACCESS.2015.2508940.

[6] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," *Stud. Comput. Intell.*, vol. 146, pp. 173–214, Apr. 2008. doi: 10.1007/978-3-540-69277-5_7.

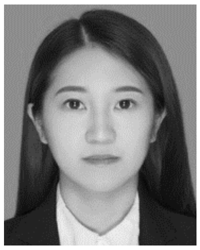
[7] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002. doi: 10.1109/71.993206.

[8] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Comput.*, vol. 17, no. 2, pp. 169–189, Jun. 2014. doi: 10.1007/s10586-013-0325-0.

[9] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 169–181, Apr./Jun. 2015. doi: 10.1109/TCC.2014.2358220.

[10] 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. doi: 10.3837/tiis.2016.03.002.
[11] B. P. Rimal and M. Maier, "Workflow scheduling in multi-tenant cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 290–304, Jan. 2017. doi: 10.1109/TPDS.2016.2556668.
[12] A. Deldari, M. Naghibzadeh, and S. Abrishami, "CCA: A deadline-constrained workflow scheduling algorithm for multicore resources on the cloud," *J. Supercomput.*, vol. 73, no. 2, pp. 756–781, Feb. 2017. doi: 10.1007/s11227-016-1789-5.
[13] 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, Mar. 2017, Art. no. e3942. doi: 10.1002/cpe.3942.
[14] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016. doi: 10.1109/TPDS.2015.2446459.
[15] Z. Li, J. Ge, H. Yang, L. Hu, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," *Future Gener. Comput. Syst.*, vol. 65, pp. 140–152, Dec. 2016. doi: 10.1016/j.future.2015.12.014.
[16] N. Anwar and H. Deng, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment," *Appl. Sci.*, vol. 8, no. 4, p. 538, 2018. doi: 10.3390/app8040538.
[17] 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. doi: 10.1155/2018/1934784.
[18] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Gener. Comput. Syst.*, vol. 83, pp. 14–26, Jun. 2018. doi: 10.1016/j.future.2018.01.005.
[19] (2019). *Amazon EC2*. [Online]. Available: http://aws.amazon.com/ec2/
[20] (2019). *Amazon EBS*. [Online]. Available: http://aws.amazon.com/ebs/
[21] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, Oct. 1988. doi: 10.1023/A:1022602019183.
[22] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," in *Proc. Int. Symp. Innov. Intell. Syst. Appl.*, Istanbul, Turkey, Jun. 2011, pp. 50–53.
[23] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Int. Conf. Parallel Problem Solving From Nature*, Paris, France, 2000, pp. 849–858.
[24] R. N. 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, Jan. 2011. doi: 10.1002/spe.995.
[25] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Chicago, IL, USA, Oct. 2012, pp. 1–8.
[26] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci.*, Austin, TX, USA, Nov. 2008, pp. 1–10.
[27] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," in *Proc. Int. Conf. Cloud Comput.*, Munich, Germany, 2009, pp. 115–131.
[28] M. A. Abido and J. M. Bakhashwain, "Optimal VAR dispatch using a multiobjective evolutionary algorithm," *Int. J. Elect. Power Energy Syst.*, vol. 27, no. 1, pp. 13–20, Jan. 2005. doi: 10.1016/j.ijepes. 2004.07.006.
[29] J. Wei and M. Zhang, "A memetic particle swarm optimization for constrained multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, New Orleans, LA, USA, Jun. 2011, pp. 1636–1643.
[30] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999. doi: 10.1109/4235.797969.
[31] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Inst. Technol., Zürich, Switzerland, 1999.
[32] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011. doi: 10.1016/j.swevo.2011.02.002.

YONGQIANG GAO received the Ph.D. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2013. He is currently an Associate Professor with the College of Computer Science, Inner Mongolia University (IMU). His research interests include cloud computing, virtualization, and green computing.



SHUYUN ZHANG received the B.S. degree from Jining Normal University, in 2017. She is currently pursuing the M.S. degree with the College of Computer Science, Inner Mongolia University. Her research interests include cloud computing and the Internet of Things.

JIANTAO ZHOU received the Ph.D. degree from Tsinghua University, China, in 2005. She is currently a Professor and a Ph.D. Supervisor with the College of Computer Science, Inner Mongolia University. Her research interests include cloud computing and software engineering.