Multiobjective Cloud Workflow Scheduling: A Multiple Populations Ant Colony System Approach

Zong-Gan Chen, *Student Member, IEEE*, Zhi-Hui Zhan, *Member, IEEE*, Ying Lin, *Member, IEEE*, Yue-Jiao Gong, *Member, IEEE*, Tian-Long Gu, Feng Zhao, Hua-Qiang Yuan, Xiaofeng Chen, *Senior Member, IEEE*, Qing Li, *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Cloud workflow scheduling is significantly challenging due to not only the large scale of workflow but also the elasticity and heterogeneity of cloud resources. Moreover, the pricing model of clouds makes the execution time and execution cost two critical issues in the scheduling. This paper models the cloud workflow scheduling as a multiobjective optimization problem that optimizes both execution time and execution cost. A novel multiobjective ant colony system based on a co-evolutionary multiple populations for multiple objectives framework is proposed, which adopts two colonies to deal with these two objectives, respectively. Moreover, the proposed approach incorporates with the following three novel designs to efficiently deal with the multiobjective challenges: 1) a new pheromone update rule based on a set of nondominated solutions from a global archive to guide each colony to search its optimization objective sufficiently; 2) a complementary heuristic strategy to avoid a colony only focusing on its corresponding single optimization objective, cooperating with the pheromone update rule to balance the search of both objectives; and 3) an elite study strategy to improve the solution quality of the global archive to help further approach the global Pareto front. Experimental simulations are conducted on five types of real-world scientific workflows and consider the properties of Amazon EC2 cloud platform. The experimental results show that the proposed algorithm performs better than both some state-of-the-art multiobjective optimization approaches and the constrained optimization approaches.**

***Index Terms*—Cloud computing, evolutionary approach, multiobjective optimization, workflow scheduling.**

Manuscript received September 5, 2017; revised February 6, 2018; accepted April 20, 2018. Date of publication May 18, 2018; date of current version May 7, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61772207 and Grant 61332002, in part by the Natural Science Foundation of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306038, in part by the Project for Pearl River New Star in Science and Technology under Grant 201506010047, in part by GDUPS (2016), in part by the Fundamental Research Funds for the Central Universities, and in part by the Science and Technology Planning Project of Guangdong Province under Grant 2015B010130002. This paper was recommended by Associate Editor S. Yang. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Z.-G. Chen, Z.-H. Zhan, Y. Lin, Y.-J. Gong, and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com; junzhang@ieee.org).

T.-L. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

F. Zhao is with the Guangxi Colleges and Universities Key Laboratory of Complex System Optimization and Big Data Processing, Yulin Normal University, Yulin 537000, China.

H.-Q. Yuan is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710126, China.

Q. Li is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2832640

I. Introduction

CLOUD computing has developed rapidly in recent years, whose computing resources (e.g., servers and storage) are accessed through network [1], [2]. The infrastructure as a service (IaaS) is the most basic and common service model that makes the cloud resources be utilized efficiently. In this way, clouds can offer a resource pool for users to lease resources and have shown promising performance on executing large-scale workflow applications [3]–[5]. Therefore, with its powerful computing capability, cloud computing has been widely applied to solve problems with massive amounts of data and complex workflows in various fields, such as biology, physics, and astronomy [6]–[10].

Workflow, containing a set of tasks with data dependence between each other, is a typical type of application on clouds. The workflow scheduling problem, which aims to find the most suitable resource for each task of the workflow to meet user defined quality of service (QoS), has been extensively researched over past years on distributed environment like grids [11]. Since the workflow scheduling is an NP-hard problem, the traditional methods such as dynamic programming or greedy algorithm are inapplicable on large scale workflow scheduling. Driven by the good performance of evolutionary computation algorithms on complex optimization problems [12]–[17], particle swarm optimization (PSO) [18] and differential evolution [19] have been proposed to deal with the workflow scheduling on grids.

However, workflow scheduling on clouds becomes more promising and popular in recent years [20], but is also more

challenging. The existing workflow scheduling approaches on traditional distributed systems like grids may be not sufficient in cloud computing environment due to the distinct features of cloud computing like elasticity, heterogeneity, and the pricing model [21]. On the one hand, unlike the static and limited resource pool of grid computing, the computing resource of cloud computing is elastic and heterogeneous. In other words, resources in cloud computing are almost unlimited for users and can be leased in any amount at any time [22], while the variety of resources is far more than grid computing. On the other hand, it would lead to unexpectedly high charges if the workflow scheduling does not take the cloud pricing model into consideration. Thus, it is quite indispensable to design an appropriative workflow scheduling approach for cloud computing platform.

Recently, increasing number of researches focused on the workflow scheduling on clouds. The workflow execution time (WET) is a common optimization objective. Raghavan *et al.* [23] proposed a novel bat algorithm and Liang *et al.* [24] applied artificial bee colony to optimize the WET on cloud. However, as cloud computing is mainly used for commercial applications, the execution cost is also an important factor for users. Pandey *et al.* [25] proposed a novel PSO-based approach to optimize the execution cost. However, the cost and the time often conflict with each other, the scheduling scheme that completes the task faster often requires a larger investment and higher cost, while a low investment often results in poor time efficiency. Therefore, it is essential to take execution time and cost into account at the same time.

Based on the considerations mentioned above, one possible solution is to model the workflow scheduling as a constrained optimization problem that considering both execution time and cost. The model that optimizes the workflow execution cost (WEC) under the constraint of deadline is highly practical since for commercial organizations, tasks need to be completed before a deadline rather than as quick as possible so that minimizing cost under deadline constraint can maximize the profit. Lin *et al.* [26] proposed a heuristic algorithm based on the partial critical paths to solve this constraint based model. Rodriguez and Buyya [27] applied PSO-based approach for the model while Chen *et al.* [28] proposed a novel dynamic objective GA (DOGA) approach. Nevertheless, it is difficult to define a deadline for the constrained optimization model since users do not know the range of the execution time in advance.

Another efficient way to consider both execution time and cost is to model the problem as a multiobjective optimization problem (MOP). Durillo and Prodan [29] extended the heterogeneous earliest-finish-time (HEFT) approach [30] to a novel multiobjective HEFT (MOHEFT) to optimize both the execution time and cost. But as an enumeration-based approach, MOHEFT suffers from very poor efficiency when dealing with large-scale workflows. Therefore, evolutionary multiobjective optimization (EMO) approaches are proposed. Zhu *et al.* [31] proposed a novel EMS-C approach based on the well-known NSGA-II framework to optimize the execution time and cost. In fact, the workflow scheduling model aims to establish a mapping from tasks to resources, which is a discrete combinatorial optimization problem. In this case, the ant colony optimization (ACO) [32]–[36] and its variants that inspired by the foraging behavior of ants are promising solvers. In our preliminary study [37], our ant colony system (ACS)-based approach to solve the constraint based cloud workflow scheduling problem showed that ACS can provide good guidance for the search of scheduling scheme owing to the pheromone and heuristic information. Therefore, this paper further studies the ACS-based EMO approach for solving the multiobjective scheduling problem.

Multiobjective ACOs are widely studied [38]–[40]. However, it is difficult to apply the existing multiobjective ACOs since the cloud workflow scheduling model has its own unique features and very few multiobjective ACOs are specifically designed for cloud workflow scheduling. In order to extend the advantages of ACS to solve the cloud workflow scheduling problem, in this paper, we adopt the efficient multiple populations for multiple objectives (MPMOs) framework [41], and propose a novel multiobjective ACS (MOACS) to solve the multiobjective cloud workflow scheduling problem.

The novelties of our MOACS approach are as follows. First, MOACS, based on the MPMO, adopts two colonies with one optimizing WET and the other optimizing WEC, being efficient to search both objectives sufficiently. Note that, for each colony, the objective it optimizes is named as optimization objective, while the other is named as external objective. Second, a new pheromone update rule based on the guidance of a set of nondominated solutions from a global archive is designed to help each colony search its optimization objective sufficiently. Third, in order to avoid a colony only focusing on its own optimization objective and performing unsatisfactorily on the external objective, a complementary heuristic strategy (CHS) that provides the guidance information of the external objective is proposed. The CHS cooperates with the pheromone update rule to balance the search of both objectives. Fourth, an elite study strategy (ESS) is performed to help the archive update process for further approaching the global Pareto front (PF). Fifth, a novel encoding scheme that reflects the elasticity and heterogeneity of the cloud computing platform is proposed. Sixth, MOACS can generate a set of scheduling schemes so that users can choose a suitable scheme according to their preference. Compare with the constrained optimization approaches, MOACS can give users more choices and better meet the QoS of users.

The rest of this paper is organized as follows. Section II presents the cloud workflow scheduling model. Section III presents the framework of MOACS and its novelties in detail. Section IV presents the experimental results and finally, Section V presents the conclusion.

II. Cloud Workflow Scheduling Model

A. Cloud Platform

Cloud platform provides computing resources usually via virtual machines (VMs) in IaaS. In this way, users can lease resources like VMs to execute the workflow tasks. There are several basic principles of cloud computing as follows. First of all, the resource on clouds is elastic, which means

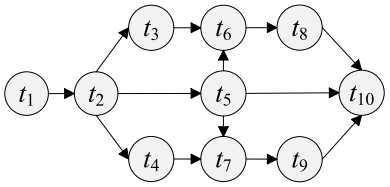

Fig. 1. Sample workflow with ten tasks.

the resource pool is nearly unlimited and users can lease resources in any amount at any time. Heterogeneity is another distinct feature that means there are various types of resource on clouds. In addition, cloud providers adopt a basic pricing model that any partial utilization of the resource is charged as a full time period. Amazon EC2 platform sets 1 h as a minimum unit of lease time while Microsoft Azure sets 1 min. The set $R = \{r_1, r_2, r_3, \ldots, r_{|R|}\}$ represents the resources that are leased for the workflow execution. We define a lease process $\mathrm{LP} = (\mathrm{LS}, \mathrm{LF}, \mathrm{LD})$ where LS represents the lease start time, LF represents the lease finish time, and LD represents the lease duration time. The calculation of the cost to lease the resource r_j for the execution of the task t_i is shown in (1), where P represents the lease price of a resource for a unit of lease time and u represents the unit of lease time

$$\mathrm{Cost}_{t_i}^{r_j} = P_{r_j} \times \left\lceil \mathrm{LD}_{t_i}^{r_j}/u \right\rceil. \tag{1}$$

B. Workflow Model

Workflow application on cloud platform with a set of interdependent tasks can be modeled as a directed acyclic graph, formulated as a tuple (T, E). $T = \{t_1, t_2, t_3, \ldots, t_{|T|}\}$ is a set of nodes representing the tasks while $E = \{\ldots, e_{ij}, \ldots\}$ is a set of edges where e_{ij} represents that t_i is the parent task of t_j, which means that t_i and t_j has data dependency. That is to say, the child task t_j cannot be executed until the end of execution of its parent task t_i. A sample workflow with ten tasks is shown in Fig. 1.

The execution time (ET) of a task is calculated by (2), where $\mathrm{fpo}(t_i)$ represents the floating point operations of t_i, which is usually used to represent the computational size of a task; $\mathrm{pc}(r_j)$ represents the processing capacity of the resource r_j. Equation (3) shows the calculation of the data transmission time between t_i and its child task t_j, where $\mathrm{datasize}(t_i, t_j)$ is the size of transmission data, *bandwidth* is the bandwidth of the network connection, while r_{t_i} and r_{t_j} are the resources that execute t_i and t_j, respectively. Note that, if the parent task and the child task are executed on the same resource, the data transmission time is ignorable and is set as 0. The data transmission time (TT) of a task, calculated by (4), is to sum the time needed for the data transmission to all its child tasks

$$\mathrm{ET}_{t_i}^{r_j} = \frac{\mathrm{fpo}(t_i)}{\mathrm{pc}(r_j)} \tag{2}$$

$$\mathrm{transmit}(t_i, t_j) = \begin{cases} \frac{\mathrm{datasize}(t_i,t_j)}{\mathrm{bandwidth}} & \text{if } (r_{t_i} \neq r_{t_j}) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\mathrm{TT}_{t_i} = \sum_{t_j \in \mathrm{child}(t_i)} \mathrm{transmit}(t_i, t_j). \tag{4}$$

TABLE I
EXAMPLE OF EXECUTION TIME BASED ON THE WORKFLOW IN FIG. 1

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
r_1	4	8	9	8	16	8	8	6	12	15
r_2	2	6	6	4	11	4	3	5	7	10
r_3	1	4	4	2	6	2	2	3	4	5

TABLE II
EXAMPLE OF TRANSMISSION TIME BASED ON THE WORKFLOW IN FIG. 1

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
t_1	0	1	0	0	0	0	0	0	0	0
t_2	0	0	3	2	1	0	0	0	0	0
t_3	0	0	0	0	0	2	0	0	0	0
t_4	0	0	0	0	0	0	1	0	0	0
t_5	0	0	0	0	0	6	1	0	0	2
t_6	0	0	0	0	0	0	0	3	0	0
t_7	0	0	0	0	0	0	0	0	2	0
t_8	0	0	0	0	0	0	0	0	0	2
t_9	0	0	0	0	0	0	0	0	0	4
t_{10}	0	0	0	0	0	0	0	0	0	0

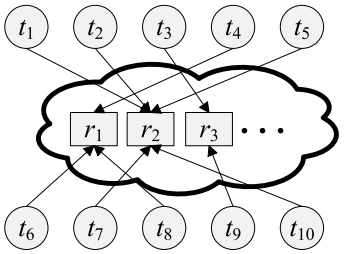

Fig. 2. Sample workflow scheduling scheme on cloud platform.

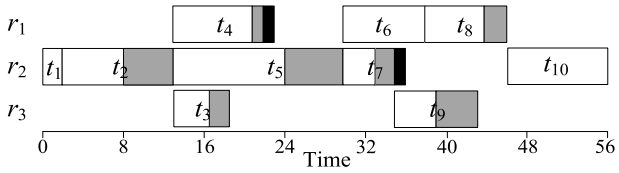

Fig. 3. Process diagram for the execution of the workflow in Fig. 1 according to the scheduling scheme in Fig. 2.

Tables I and II show examples of execution time and data transmission time, respectively, based on the workflow in Fig. 1. In Table II, if there is no data dependency between two tasks, the data transmission time is 0.

C. Workflow Scheduling

Workflow scheduling on clouds is to schedule the tasks of the workflow to the resources on the cloud platform. In essence, workflow scheduling establishes a mapping between tasks and resources. Fig. 2 shows a sample scheduling scheme of the workflow in Fig. 1 that assigns the tasks to the cloud resources.

With the data in Tables I and II, Fig. 3 shows a process diagram for the execution of the workflow according to the scheduling scheme in Fig. 2. In the figure, white grids represent the execution time while those gray grids represents the data transmission time. We can see that t_1 and t_2 have data dependency but they are executed on the same resource

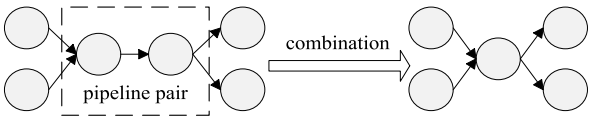

Fig. 4. Example of pipeline pair tasks.

Encoded solution

dim	1	2	3	4	5	6	7	8	9	10
value	2	4	3	1	4	5	2	1	3	4

Fig. 5. Example of encoded solution.

so that no data transmission time is needed. Similarly, no data transmission time is needed between t_6 and t_8. In this example, assume that the unit of lease time is 2. The tasks t_4 and t_7 only need a part of a unit lease time for executing. But according to the basic pricing model of cloud computing, a full unit lease time should be charged. Therefore, the black grids are used to represent the idle time caused by the basic pricing model.

III. MOACS Approach

The MOACS approach is based on the ACS optimizer and the MPMO framework that uses multiple colonies to optimize different objectives. A global archive is built to store the nondominated solutions during the search process. Moreover, we propose a new pheromone update rule based on the global archive, and a novel CHS that provides guidance information of the external objective for the colonies. Lastly, an ESS is adopted during the archive update process to help further approach the global PF. The complete MOACS approach is described as follows.

A. Solution Encoding

Before the solution encoding, two preprocessing operations are executed. The first operation is "pipeline pair" tasks combination proposed in [42]. As shown in Fig. 4, pipeline pair is a special pair of a parent task and its child task, where the parent task only has one child task while the child task only has one parent task. After this combination operation, the pipeline pair tasks are treated as one task and will be scheduled on the same resource. This operation can reduce the search space without breaking the entire topology structure. The second operation is to carry out a topological sort, which is performed on the $|T|$ tasks after the pipeline pair tasks combination operation. This operation is to sort the tasks according to their parental–child relationship topology and assign each task with an index ranging from 1 to $|T|$ according to the sorting results.

A solution of MOACS, defined as sol, is a sequence that encoded by the indexes of the tasks. Fig. 5 shows an example of encoded solution. The dimension i represents the task t_i and its corresponding value sol[i] is the index of the resource, meaning that t_i is scheduled on the $r_{\text{sol}[i]}$. According to this principle, t_4 and t_8 are scheduled on r_1; t_1 and t_7 are scheduled on r_2; and so on. The length of a solution is equal to $|T|$ while the range of each dimension depends on the scale of the resource pool. In defining the search range, the elasticity and heterogeneity of cloud resources should be considered.

On the one hand, the resource pool in cloud computing is elastic and nearly unlimited for users, but it is quite impossible to deal with such an infinite search space. A feasible solution is to adopt a resource pool with fixed number of resource. However, it is also difficult to define the scale of the resource pool. If the resource pool is too small, the elasticity of cloud computing could not be well reflected while a large resource pool will result in the expansion of the search space, which will slow down the convergence and increase the difficult to find a suitable solution.

In MOACS, we use the MPT to represent the maximal parallel tasks of the workflow, which means that at most MPT tasks are run in parallel during the workflow execution. Under the most extreme condition, these MPT tasks are elastically scheduled on MPT resources of the same type. Therefore, we have to assume that one type of resource has MPT available resources, which can reflect the elasticity of cloud computing and meanwhile reduce the search space as much as possible. A naïve way to estimate the value of MPT is to set it as $|T|$ that all the tasks are executed in parallel. However, due to the constraint of the workflow's topology structure, all tasks cannot be executed in parallel. Therefore, we use the following method to calculate MPT. First, a special scheduling scheme is defined by leasing a new resource for each task to encourage parallelization as much as possible. In other words, $|T|$ resources are leased for the $|T|$ tasks, respectively. Second, we simulate the workflow execution process according to this scheme and obtain the execution process (e.g., the start time and finish time) of each task. Finally, according to the start time and finish time of each task, we can obtain the maximal number of parallel tasks of this scheduling scheme. This number is set as the MPT value of the workflow.

On the other hand, we define *rtype* different types of resources to reflect the heterogeneity of cloud computing. As a result, the scale of the resource pool, defined as $|R_{\text{pool}}|$, is MPT $\times$ rtype, where $r_1 \sim r_{\text{MPT}}$ are the first type of resource, $r_{\text{MPT}+1} \sim r_{\text{MPT}\times 2}$ are the second type of resource, and so on.

B. Fitness Evaluation

Two optimization objectives are considered that are the WET and the WEC, which are formulated as

$$\text{minmize } f = (\text{WET}, \text{WEC})^T. \tag{5}$$

The procedure of calculating WET and WEC based on an encoded solution is shown in Fig. 6. First, we initialize the set of leased resource R as Ø. Then we iterate every task to simulate the workflow execution. For a task t_i of the workflow, we can obtain the resource it executed on, named as r_{t_i}, from the encoded solution. If r_{t_i} has not been leased, add r_{t_i} to R and initialize the resource free time (RFT) of r_{t_i} as 0. The execution of t_i should wait until its target resource is free. In addition, on the basis of the workflow's topology structure, if t_i has parent tasks, it should also wait until its parent tasks finish their executions. Therefore, the begin time (BT) of t_i is equal

Procedure *Calc_WET_WEC*
  $R = \varnothing$;
  for i=1 **to** i=$|T|$
    if $r_{t_i} \notin R$
      $R = R \cup \{r_{t_i}\}$; $RFT_{r_{t_i}} = 0$;
    end if
    $BT_{t_i} = RFT_{r_{t_i}}$;
    if t_i has parents tasks
      for each $t_{par} \in parents(t_i)$
        $BT_{t_i} = \max\{BT_{t_i}, FT_{t_{par}}\}$;
      end for
    end if
    $ET_{t_i}^{r_{t_i}} = \frac{fpo(t_i)}{pc(r_{t_i})}$; $TT_{t_i} = \sum_{t_j \in child(t_i)} transmit(t_i, t_j)$;
    $TPT_{t_i} = ET_{t_i}^{r_{t_i}} + TT_{t_i}$; $RFT_{r_{t_i}} = FT_{t_i} = BT_{t_i} + TPT_{t_i}$;
    $add(TLP_{r_{t_i}}, LP(BT_{t_i}, FT_{t_i}, TPT_{t_i}))$;
  end for
  $WET = WEC = 0$;
  for i=1 **to** i=$|T|$
    $WET = \max(WET, FT_{t_i})$;
  end for
  for j=1 **to** j=$|R|$
    $MergeAdjacentLeaseProcess(TLP_{r_j})$;
    for each LP_{r_j} **in** TLP_{r_j}
      $WEC += P_{r_j} \times \lceil LD_{LP_{r_j}} / u \rceil$;
    end for
  end for
End procedure

Fig. 6. Procedure of calculating WET and WEC.

to the maximum of the RFT of r_{t_i} and the finish time of all its parent tasks. The execution time and data transmission time of t_i are calculated according to (2) and (4), respectively. TPT is defined as the total processing time of a task, which is the summarization of execution time and data transmission time. After that, the finish time (FT) of t_i is obtained by summing up the begin time and the total process time of t_i. Then the RFT of r_{t_i} is updated as the finish time of t_i because r_{t_i} is occupied until the execution of t_i finishes. A vector, named as total lease process (TLP), is defined to store all the lease process on a resource. The execution of t_i creates a new lease process that the lease start time and the lease finish time are equal to the begin time and the finish time of t_i, respectively, while the lease duration is the total process time of t_i. We add this new lease process to the total lease process of r_{t_i} for calculating WEC later.

After applying the scheduling scheme to simulate the workflow execution, we initialize WET and WEC as 0. WET is the finish time of the total workflow, which is equal to the finish time of the last task. In detail, WET is the maximum of the finish time of all tasks of the workflow. In Fig. 3, some tasks are executed on the same resource successively. In such case if directly using (1) to calculate the cost of every task's lease process, the cost may be inaccurate because some idle time may actually be exploited. For example in Fig. 3, 1 idle time is charged if using (1) to calculate the cost of t_2's lease process because the cloud computing providers adopt a basic pricing model that the lease of resource has a minimum unit of time. But actually, t_5 is executed successively after the execution of t_2 and utilizes this part of time. Therefore, the duration of t_5's execution also contain the part that is considered as idle time in t_2's lease process so that this part of time is charged again. In order to deal with this problem, we first apply a *MergeAdjacentLeaseProcess* operation to merge the adjacent lease processes on the resource r_j before calculating the lease cost of r_j. In more detail, if the start time of a lease process is before another lease process on the same resource finishing and running out a complete lease unit, we merge these two lease processes. Finally, WEC is the sum of the lease charge of all leased resources.

C. Multiple Colonies Framework

Since it is difficult to assign a suitable fitness value for an individual under an MOP model, MOACS adopts the concept of the MPMO technique proposed by Zhan *et al.* [41], which treats different optimization objectives separately in different populations. Therefore, two colonies with the same number of ants are used. These two colonies are named "time colony" *Tcolony* and "cost colony" *Ccolony*, where *Tcolony* sets WET as optimization objective while *Ccolony* sets WEC as optimization objective. Under the MPMO framework, both colonies work similar as the traditional ACS to construct solutions by using pheromone and heuristic information, which will be described in Section III-D. However, *Tcolony* and *Ccolony* use their separate pheromone and heuristic information respectively during solution construction.

D. Solution Construction

During the evolutionary process, ants construct their solutions in parallel within their corresponding colonies. In each step, each ant selects a resource for a task according to the pheromone τ and the heuristic information η. The solution construction process is shown as (6) and is described as follows. For the task t_i, a random number q ranging in [0, 1] is first generated. If $q \leq q_0$, ant exploits greedily to select the resource with the highest pheromone and heuristic information, denoted by the resource that has the largest $[\tau(i,j)] \times [\eta(i,j)]^{\beta}$ value where $\tau(i,j)$ and $\eta(i,j)$ represent the pheromone and heuristic information deposited between the task t_i and the resource r_j, respectively, and β is a parameter. Otherwise, ant selects the resource for t_i by roulette wheel selection according to the probability defined in (7). The designs of pheromone and heuristic information are described in Sections III-E and III-F, respectively,

$$r_{t_i} = \begin{cases} \text{argmax}\left\{\left[\tau(i,j)\right] \times \left[\eta(i,j)\right]^{\beta}\right\} & \text{if } q \leq q_0 \\ \text{Roulette Wheel Selection} & \text{otherwise} \end{cases} \tag{6}$$

$$p(i,j) = \frac{\left[\tau(i,j)\right] \times \left[\eta(i,j)\right]^{\beta}}{\sum_{k=1}^{|R_{\text{pool}}|} [\tau(i,k)] \times [\eta(i,k)]^{\beta}}. \tag{7}$$

E. Pheromone Update

1) Initialization: In MOACS, the initial pheromone value τ_0 of *Tcolony* and *Ccolony* are defined as (8) and (10), respectively. In *Tcolony*, the time greedy solution (TGS) is generated by (9) that selects the most efficient (fastest) resource for each

task. In *Ccolony*, the cost greedy solution (CGS) is generated by (11) that selects the most economical (cheapest) resource

$$\text{Tcolony: } \tau_0 = 1/(|T| \times \text{WET}_{\text{TGS}}) \tag{8}$$

$$\text{TGS}[i] = \arg\min\left(\text{ET}_{t_i}^{r_j}\right) j = \left\{1, 2, \ldots, \left|R_{\text{pool}}\right|\right\} \tag{9}$$

$$\text{Ccolony: } \tau_0 = 1/(|T| \times \text{WEC}_{\text{CGS}}) \tag{10}$$

$$\text{CGS}[i] = \arg\min\left(\text{Cost}_{t_i}^{r_j}\right) j = \left\{1, 2, \ldots, \left|R_{\text{pool}}\right|\right\}. \tag{11}$$

2) Local Update: The local update of pheromone occurs during the solution construction process. The local update rule is shown as

$$\tau(i,j) = (1-\rho) \times \tau(i,j) + \rho \times \tau_0 \tag{12}$$

where ρ is a parameter. When an ant selects r_j to execute t_i, the pheromone $\tau(i,j)$ is updated immediately. From (8) and (10), we can see that τ_0 is equal or less than $\tau(i,j)$ in most cases so that the pheromone $\tau(i,j)$ often decreases by using (12), which reduces the probability that other ants still schedule t_i to r_j. Thus other ants will be more likely to schedule t_i to the other resources different from r_j. The local update of pheromone can greatly improve the search diversity and avoid premature convergence. Since the local update is only adopted to evaporate the pheromone to enhance the search diversity and is not related to colonies' optimization objective, two colonies follow the same local update rule as (12). However, it should be noted that the τ_0 in two colonies are different, which are defined in (8) and (10), respectively.

3) Global Update: In traditional single objective ACS, the historically best solution is selected for the global update of pheromone. However, when deal with MOPs, there are a set of nondominated solutions in the global archive that contain the elite experience (see Section III-G for the global archive), which may all be able to provide useful guidance for the search process. Some researchers proposed to randomly select a nondominated solution from the archive [41]. Random selection can well maintain the diversity of the population but it may also lead to poor efficiency of search guidance.

Since *Tcolony* and *Ccolony* minimize WET and WEC, respectively, the solutions with low value of its optimization objective can provide useful guidance. But if we directly select the solution with the smallest value of the optimization objective for the pheromone global update, which means that *Tcolony* and *Ccolony* select the solution with the smallest WET and WEC from the global archive, respectively, it may easily be trapped into local optima and cause premature convergence.

Therefore, random selection may lack of search guidance while totally greedy selection may cause premature convergence. In order to balance the search efficiency and the diversity, we propose the procedure shown in Fig. 7 to select a solution for pheromone global update and define this solution as global update solution (GUS). First, the solutions in the global archive are sorted in ascending order according to the WET value and the number of solutions in the global archive is recorded as K. Then a selection rate sr in range of (0, 1) is defined. A random solution from the first $K \times$ sr solutions of the global archive, which has small WET, is selected as the GUS for *Tcolony* while a random solution from the last $K \times$ sr solutions, which has small WEC, is selected as the GUS for *Ccolony*. The sr is used to control the diversity of the population and guarantee the convergence speed of MOACS.

Procedure *GUS_selection*
  ascending_sort(*globalArchive*,*WET*);
  K = sizeof(*globalArchive*);
  *ind*1 = random(1, $K \times sr$);
  globalArchive[*ind*1] is selected as *Tcolony*'s *GUS*;
  *ind*2 = random($K \times (1-sr)+1, K$);
  globalArchive[*ind*2] is selected as *Ccolony*'s *GUS*;
End procedure

Fig. 7. Procedure of GUS selection.

After selecting the GUS, the pheromone global update is conducted according to (13), where ε is the pheromone enhancement parameter and the calculation of $\Delta\tau_b(i,j)$ is shown in (14) and (15) for *Tcolony* and *Ccolony*, respectively. Note that the GUS of these two colonies are not the same. The global update is only conducted on the colony's selected GUS solution. After the pheromone global update, the pheromone on the selected GUS increases and in this way the knowledge for the optimization objective accumulates

$$\tau(i,j) = (1-\varepsilon) \times \tau(i,j) + \varepsilon \times \Delta\tau_b(i,j), \forall(i,j) \in \text{GUS} \tag{13}$$

$$\text{Tcolony}: \Delta\tau_b(i,j) = 1/\text{WET}_{\text{GUS}} \tag{14}$$

$$\text{Ccolony}: \Delta\tau_b(i,j) = 1/\text{WEC}_{\text{GUS}}. \tag{15}$$

F. Complementary Heuristic Strategy

With the update of pheromone, the knowledge for *Tcolony* and *Ccolony* to optimize their separate optimization objective continually accumulates so that *Tcolony* and *Ccolony* perform well in optimizing WET and WEC, respectively. However, if a colony focuses totally on its optimization objective, it suffers from the poor performance on the external objective. That is to say, *Tcolony* may find those solutions with small WET but very high WEC while *Ccolony* may find those with small WEC but very high WET, which locate on only the margins of the PF. To address this problem, we propose the CHS as in (16) and (17) that provides guidance information of the external objective, which will be used during the solution construction, i.e., used in (6) and (7). Specially, the heuristic information for assigning task t_i on resource r_j in *Tcolony* considers the execution and data transmission cost, as

$$\eta_{\text{Tcolony}}(i,j) = \frac{1}{P_{r_j} \times \text{ET}_{t_i}^{r_j} + \sum_{t_k \in \text{parent}(t_i)} \left(P_{r_{t_k}} \times \text{transmit}(t_k, t_i)\right)} \tag{16}$$

while the heuristic information for assigning task t_i on resource r_j in *Ccolony* considers the execution and data transmission time, as

$$\eta_{\text{Ccolony}}(i,j) = \frac{1}{\text{ET}_{t_i}^{r_j} + \sum_{t_k \in \text{parent}(t_i)} \text{transmit}(t_k, t_i)}. \tag{17}$$

Herein, we use the reciprocal in (16) and (17) so that smaller cost and time will result in larger heuristic information values, which are more preferable according to (6) and (7) in the solution construction. One notice is that the heuristic information for assigning task t_i on resource r_j may be different in different ants because their scheduling scheme for t_i's parent tasks may be different. Therefore, each ant calculates its own heuristic information for assigning task t_i on resource r_j.

With such a complementary heuristic information guidance during solution construction, colonies can optimize their own optimization objective while ensuring the quality of the external objective.

G. Global Archive

During the evolutionary process of MOACS, we adopt a global archive, defined as *globalArchive*, to store those nondominated workflow scheduling solutions that have been found by ants from both *Tcolony* and *Ccolony*. The global archive collects the elite experiences, which can also help the pheromone global update as described in Section III-E.

1) Archive Initialization: In order to help pheromone have a good guidance in the beginning of the search, we initialize the *globalArchive* with some predefined solutions. Two special types of solutions are defined. For the first type, a solution is to schedule all the tasks on only one resource of a specific type. As there are *rtype* types of resource, *rtype* predefined scheduling solutions are generated. For the second type, a solution is to schedule all the tasks on the resources of the same type, which means for every type of resource, a solution is constructed by randomly selecting a resource of this type for each task. There are also *rtype* predefined scheduling solutions generated for this case. In addition, the solution found by HEFT are also added to *globalArchive*. Therefore, there are totally $2 \times \text{rtype} + 1$ predefined scheduling solutions generated and added to *globalArchive* at the beginning of the MOACS approach.

2) Archive Update: During an evolutionary generation, once the solution construction processes in both *Tcolony* and *Ccolony* have finished, the *globalArchive* is updated. First, we calculate the WET and WEC of the solutions that obtained by the ants in the two colonies. The next step is to add the solutions from *Tcolony* and *Ccolony* to *globalArchive* and then eliminating those dominated solutions from *globalArchive*. Lastly, an ESS is applied on *globalArchive*. Herein, the ESS applied on *globalArchive* is to improve the quality of solutions in *globalArchive* so as to help further approach the global PF. The motivation and process of the ESS are described as follows.

3) Elite Study Strategy: The nondominated solutions that carry the elite knowledge are collected in *globalArchive* during the evolutionary process, which have great potentials to generate good solutions. Therefore, we propose an ESS to deal with the solutions in *globalArchive* to help further approach the global PF. The procedure of ESS is shown in Fig. 8. Herein, we define two kinds of ESS to generate new solution from an elite solution in *globalArchive*. The first one is small-scope ESS, which is to randomly select a task t_{ind1} and a new solution is generated by randomly selecting a new resource for t_{ind1}. The small-scope ESS searches around an elite solution to exploit the local area of the PF. The second one is large-scope ESS, which is to randomly select a task's corresponding resource r_{ind2}, and a new solution is generated by randomly selecting a new resource that has a different type from r_{ind2}, and all tasks that are on r_{ind2} are scheduled on the new resource. The large-scope ESS enhances the global search ability, exploring the other area of the PF. These two ESS methods cooperatively help further approach the global PF.

```
Procedure EliteStudyStrategy
  if (sizeof(globalArchive) ≤ ess_num)
    selectedArchive = globalArchive;
  else{
    descending_sort(globalArchive, crowd_distance);
    selectedArchive = first ess_num solutions in globalArchive;
  }
  for each sol ∈ selectedArchive{
    newsol = sol;
    randnum = random(0,1);
    if (randnum < ess_rate){
      //the small-scope ESS
      ind1 = random(1,|T|);
      newsol[ind1] = random(1,|R_pool|);
    }
    else{
      //the large-scope ESS
      ind2 = sol[random(1,|T|)];
      r_newres = a random resource of different type from r_ind2;
      for each t_i ∈ T and t_i's resource is r_ind2
        newsol[i] = newres;
    }
    Calc_WET_WEC(newsol);
    add newsol into globalArchive;
  }
  eliminate_dominated(globalArchive);
End procedure
```

Fig. 8. Procedure of ESS.

Since it is quite inefficient to apply the ESS for all solutions in *globalArchive*, particularly when the *globalArchive* is large. Therefore, if the size of *globalArchive* exceeds a parameter *ess_num*, we first sort the *globalArchive* in descending order according to the crowding distance [43] and select the first *ess_num* solutions from *globalArchive*. Each selected solution selects one of the two ESSs according to a predefined probability *ess_rate*. The next step is to calculate the WET and WEC of the new solution and add it into the *globalArchive*. Lastly, the dominated solutions in *globalArchive* are eliminated and a new *globalArchive* is formed for the next generation.

H. Flowchart of MOACS

Fig. 9 shows the flowchart of the MOACS approach. At first, the global archive is initialized. Then two colonies named *Tcolony* and *Ccolony* run in parallel. These two colonies have independent pheromone and heuristic information. Moreover, the solution construction processes of these two colonies are

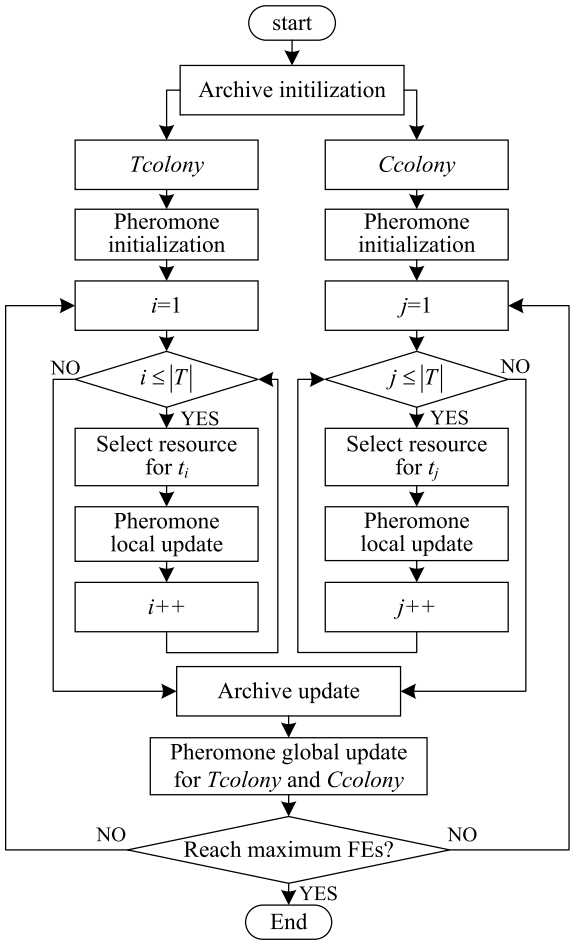

Fig. 9. Flowchart of MOACS.

independent. After the pheromone initialization for *Tcolony* and *Ccolony* by (8) and (10), respectively, the evolution begins. In every generation, ants construct their solutions in parallel by (6) and (7). During the solution construction process, the CHS is adopted and the pheromone local update is carried out. After all ants have finished the solution construction, the archive update process is conducted. Note that the ESS is included in the archive update process. Then the pheromone global update for *Tcolony* and *Ccolony* is carried out. Also note that although *Tcolony* and *Ccolony* select their GUS from the same global archive, their update processes are independent within the two colonies. MOACS terminates until reaching a maximum function evaluations, and output a set of nondominated solutions in the global archive and their corresponding WET and WEC.

I. Complexity Analysis

The time complexity of MOACS is related to the number of evolutionary generations G, the colony size N, the global archive size K, the number of tasks $|T|$, and the scale of the resource pool $|R_{\text{pool}}|$. For each ant during the solution construction in an evolutionary generation, calculating a task t_i's heuristic information [i.e., $\eta(i,j)$], according to (16) and (17), need to scan t_i's parent tasks, so the time complexity is $O(|T|)$. In addition, the $[\tau(i,j)] \times [\eta(i,j)]^{\beta}$ value should be calculated for each resource with the time complexity of $O(|R_{\text{pool}}|)$ when scheduling t_i according to (6) and (7). Therefore, an ant's solution construction for all the tasks needs the time complexity of $O(|T|^2|R_{\text{pool}}|)$ and therefore the overall time complexity of a colony's solution construction in each generation is $O(N|T|^2|R_{\text{pool}}|)$. Note that the time complexities of the solution construction in two colonies are the same. Moreover, in each generation, the crowding distance based sorting and the process of eliminating dominated solutions in archive update have the same time complexity as $O(K \log K)$. Also, the pheromone global update requires a sorting of the global archive so the time complexity is also $O(K \log K)$. Therefore, the overall time complexity of MOACS in all the G generations is $O(GN|T|^2|R_{\text{pool}}| + GK \log K)$.

The space complexity of MOACS is measured by the storage of pheromone, heuristic information, and the global archive. The pheromone is deposited between each task and each resource, which is shared by all the ants in the same colony, so the space complexity of the pheromone in two colonies is $O(2|T||R_{\text{pool}}|)$. The heuristic information is also deposited between each task and each resource. However, each ant stores its own heuristic information, so the space complexity of the heuristic information in two colonies is $O(2N|T||R_{\text{pool}}|)$. The global archive has K solutions and each solution is a $|T|$-length sequence, so the space complexity of the global archive is $O(K|T|)$. Therefore, the overall space complexity of MOACS is $O(2|T||R_{\text{pool}}|+2N|T||R_{\text{pool}}|+K|T|)$, which can be reduced to $O(N|T||R_{\text{pool}}| + K|T|)$.

TABLE III
FEATURES OF THE RESOURCE ON AMAZON EC2

Type	Processing capacity (MFLOPS)	Bandwidth (bytes/second)	Lease cost ($/hour)
m1.small	4400	39,321,600	0.06
m1.medium	8800	85,196,800	0.12
m1.large	17600	85,196,800	0.24
m1.xlarge	35200	131,072,000	0.48
m3.xlarge	57200	131,072,000	0.50
m3.2xlarge	114400	131,072,000	1.00

IV. EXPERIMENTS AND COMPARISONS

A. Experimental Environment

The experiments are conducted on six different types of resource on the current Amazon EC2 cloud platform. The processing capacity of a resource is represented in million floating point operations per second (MFLOPS) according to the number of EC2 computing units that estimated by the previous research of Ostermann *et al.* [44]. The lease unit of time is set as 1 h, the same as the Amazon EC2 cloud platform. The detailed features of the six types of resource are listed in Table III.

Five different types of real-world workflows, Montage, Epigenomics, CyberShake, LIGO Inspiral Analysis, and SIPHT, are applied in the experiments. These workflows have different characteristics and are widely used to evaluate the performance of the workflow scheduling approaches. Fig. 10 shows the structures of these five workflow types. More details of these workflows can be referred to [45] and [46].

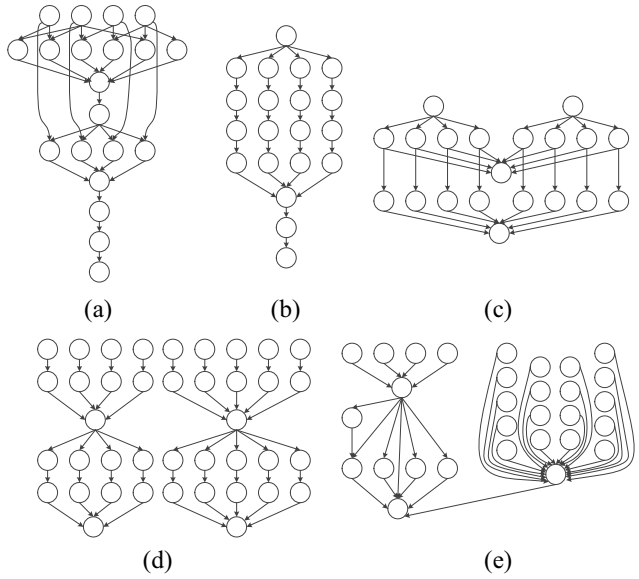

Fig. 10. Structures of the real-world workflows used for the experiments. (a) Montage. (b) Epigenomics. (c) CyberShake. (d) LIGO inspiral analysis. (e) SIPHT.

B. Experimental Settings

The following experiments are divided into three parts. We first compare MOACS with HEFT [30] about the capacity to optimize WET. Second, we compare MOACS with five multiobjective optimization approaches, including ECMSMOO [47], MOHEFT [29], NSGA-II [43], EMS-C [31], and MODE [19]. Among them, NSGA-II is a classic EMO framework. MODE is originally designed for the grid computing platform while the other three are recently proposed approaches well-designed for cloud computing platform. Third, constrained optimization model is another common method to solve the cloud workflow scheduling problem, with some well performed approaches based on PSO [27] and DOGA [28] that set WET as the constraint and WEC as optimization objective in the literature. Therefore, we compare MOACS with these two approaches.

The parameter configurations of the compared algorithms are all based on the suggestions in the corresponding references. The details of the parameter setting are shown as follows. HEFT and MOHEFT are heuristic approaches and do not need a population while the population size of the other compared approaches are all 50.

In ECMSMOO, inertia weight ω linearly decreases from 0.9 to 0.4 with the generation increases, acceleration coefficients is set as $c_1 = c_2 = c_3 = c_4 = 1$. In MOHEFT, the number of tradeoff solutions is set as 50. In NSGA-II, the simulated binary crossover operator and polynomial mutation operator in continuous space search are not suitable to solve the multiobjective cloud workflow scheduling model. Therefore, the one-point crossover and random mutation are adopted instead. The crossover rate is set as 1 and the mutation rate is set as $1/|T|$, where $|T|$ represents the number of tasks in the workflow due to their good performance in the parameter test. In EMS-C, the crossover rate and mutation rate are set as 1 and $1/|T|$, respectively. In PSO, the inertia weight $\omega = 0.5$ and acceleration constants $c_1 = c_2 = 2.0$. In DOGA, the crossover rate $P_c = 0.8$ and the mutation rate $P_m = 0.002$ during the process to meet deadline constraint, while $P_c = 0.15$ and $P_m = 0.008$ during the process to minimize WEC.

TABLE IV
COMPARISONS OF MOACS WITH HEFT ON WET

Workflow	(*WET* (HEFT) / *WET* (MOACS) − 1) ×100%	Workflow	*WET* (HEFT) / *WET* (MOACS) − 1) ×100%
CyberShake 30	**62.49%**	CyberShake 100	**71.33%**
CyberShake 50	**42.77%**	CyberShake 1000	**124.26%**
Epigenomics 24	**0.40%**	Epigenomics 100	**0.25%**
Epigenomics 46	**0.22%**	Epigenomics 997	**15.51%**
Inspiral 30	**0.15%**	Inspiral 100	**49.53%**
Inspiral 50	**0.22%**	Inspiral 1000	**144.46%**
Montage 25	**29.71%**	Montage 100	**0.15%**
Montage 50	**18.64%**	Montage 1000	**301.89%**
SIPHT 30	**14.74%**	SIPHT 100	**18.73%**
SIPHT 60	**16.47%**	SIPHT 1000	**36.23%**

ACOs do not need a large population that the population size set as 10 is commonly used [33]. Therefore, the *Tcolony* and *Ccolony* each has 5 ants in our MOACS approach. In addition, β is 5 while q_0 is 0.9. The parameter ρ in pheromone local update and the parameter ε in pheromone global update are set to 0.1. The selection rate *sr* during the pheromone global update is set to 0.1. The *ess_num* and *ess_rate* during the archive update are set to 30 and 0.2, respectively.

For fair comparison, the maximum function evaluations (the procedure to calculate WET and WEC of a scheduling scheme) is set to 60 000. In order to avoid the stochastic influence, ten independent runs are conducted except HEFT and MOHEFT since they are not evolutionary approaches.

C. Experimental Results

1) Comparison With Non-Metaheuristic HEFT: HEFT is an effective single-objective non-metaheuristic approach to optimize WET. As MOACS is a multiobjective approach, it obtains a set of solutions with both WET and WEC in each run. Herein, we use the solution with minimal WET in each run and calculate their average value for comparison. Table IV shows the comparison of MOACS with HEFT. The results are represented by $(\mathrm{WET(HEFT)}/\mathrm{WET(MOACS)} - 1) \times 100\%$, meaning the improvement percentage of MOACS than HEFT in optimizing WET. The results show that the improvement is slight in some small-scale workflows, such as Epigenomics 46 and Inspiral 30. However, as the workflow scale becomes large, the improvement of MOACS becomes much significant, particularly in the Inspiral 1000 and Montage 1000 cases. This may be due to that HEFT is a greedy heuristic so it may easily fall into local optima in large scale problems, while MOACS maintains diversity well to explore the search space.

2) Comparison With Multiobjective Optimization Approaches: Since the cloud workflow scheduling problem is a real-world application, we do not know the true PF in advance. As a result, the popular performance metric like inverted generational distance indicator is not applicable in the experiments. Therefore, we adopt two other performance metrics, hypervolume (HV) and $C(A, B)$ [48], to evaluate the performance of these six approaches.

TABLE V
COMPARISONS OF MOACS WITH THE OTHER MULTIOBJECTIVE OPTIMIZATION APPROACHES ON HYPERVOLUME [(HV (MOACS)/HV (OTHER APPROACHES) − 1)×100%]

Workflow	ECMSMOO	MOHEFT	NSGA-II	EMS-C	MODE
CyberShake 30	**28.71%**	**0.64%**	**20.98%**	**0.10%**	**67.79%**
CyberShake 50	**38.38%**	**0.43%**	**26.41%**	**1.05%**	**67.46%**
CyberShake 100	**42.43%**	**1.07%**	**34.54%**	**1.11%**	**92.80%**
CyberShake 1000	**54.33%**	**N/A**	**192.58%**	**1.41%**	**1132.76%**
Epigenomics 24	**23.38%**	**1.41%**	**14.46%**	−0.023%	**53.08%**
Epigenomics 46	**46.07%**	**18.64%**	**11.79%**	**0.08%**	**44.52%**
Epigenomics 100	**60.30%**	**31.26%**	**29.06%**	**0.38%**	**115.96%**
Epigenomics 997	**87.52%**	**N/A**	**78.33%**	**1.44%**	**988.68%**
Inspiral 30	**26.85%**	**1.55%**	**22.87%**	**0.20%**	**62.67%**
Inspiral 50	**41.74%**	**3.90%**	**31.53%**	**0.86%**	**86.76%**
Inspiral 100	**26.70%**	**1.03%**	**22.29%**	**0.25%**	**43.05%**
Inspiral 1000	**59.68%**	**N/A**	**78.97%**	**2.61%**	**307.08%**
Montage 25	**27.26%**	**0.39%**	**24.68%**	**0.23%**	**57.00%**
Montage 50	**32.88%**	**0.27%**	**26.13%**	**0.41%**	**46.96%**
Montage 100	**38.65%**	**0.12%**	**36.12%**	**0.29%**	**56.68%**
Montage 1000	**85.18%**	**N/A**	**91.83%**	**2.82%**	**312.18%**
SIPHT 30	**47.78%**	**0.092%**	**7.47%**	−0.004%	**49.81%**
SIPHT 60	**56.86%**	**0.78%**	**14.93%**	**0.65%**	**57.93%**
SIPHT 100	**64.36%**	−0.026%	**20.50%**	**0.75%**	**72.78%**
SIPHT 1000	**23.49%**	**N/A**	**22.67%**	**0.69%**	**62.37%**

HV represents the diversity and convergence by calculating the volume among a set of solutions and a reference point. A larger HV is preferable as it represents better quality and distribution of the obtained solutions. We first combine the solutions that all the approaches found in 10 runs into a set. Then the combination of the worst objective values of WET and WEC (the highest WET and WEC) among all these solutions, is selected as the reference point. The HV of the solution set obtained in 10 runs is calculated independently and the average HV value among 10 runs is reported.

$C(A, B)$ compares the dominance relationship between two solution sets. $C(A, B)$, calculated by (18), represents the ratio of solutions in B that are dominated by solutions in A. The range of $C(A, B)$ is within [0, 1]. When $C(A, B)$ is 0, it means that all solutions found by B is not dominated by any solution found by A. When $C(A, B)$ is 1, it represents that for each solution found by B, there are at least one solution found by A that dominate or equal to it. All nondominated solutions found in 10 runs are collected to calculate the $C(A, B)$ value as

$$C(A, B) = \frac{|\{b \in B, \exists a \in A, a \text{ dominates or equal to } b\}|}{|B|}. \tag{18}$$

Table V shows the comparisons of MOACS with ECMSMOO, MOHEFT, NSGA-II, EMS-C, and MODE on HV. The results are represented by (HV (MOACS)/HV (other approaches) − 1) × 100%, which means the improvement percentage of MOACS than the other five approaches. A positive number represents MOACS is better. From Table V, we can see that the performance of MOACS is far better than ECMSMOO, NSGA-II, and MODE. Since MOHEFT has very high time complexity so that it cannot finish execution in an acceptable time for large-scale workflows, we mark them as "*N/A*." MOACS performs better than MOHEFT in most of the workflows except SIPHT100 where MOHEFT is slightly better, while in some workflows such as Epigenomics 100 and Inspiral 50, the advantage of MOACS is very obvious. Moreover, in the large-scale workflows, MOACS performs better distinctly, particularly in Inspiral 1000 and Montage 1000.

Table VI shows the comparison results of $C(A, B)$. We *bold* those C(MOACS, –) higher than 90%, which means the obtained set of MOACS dominates or equal to most of the solutions obtained by the other approaches, and those C(–, MOACS) less than 10%, which means that the obtained set of the other approaches can only dominate or equal to very few solutions obtained by MOACS. Compare with ECMSMOO, NSGA-II, and MODE, the values of C(MOACS, –) and C(–, MOACS) in most cases are bolded and many of them are 100% and 0, respectively, which means that the obtained set of MOACS totally dominates those of ECMSMOO, NSGA-II, and MODE. It is interesting that in some cases such as CyberShake 50, the C(MOACS, EMS-C) value is smaller than C(EMS-C, MOACS) but the HV value of MOACS is better. That is because $C(A, B)$ metric only shows the dominance relationship, but does not show the "intensity" of the dominance. In detail, those solutions of EMS-C that dominate MOACS's just have slight improvement but in turn, those solutions of MOACS that dominates EMS-C's have great advantage.

In order to observe the performance intuitively, we illustrate the solutions found by ECMSMOO, MOHEFT, NSGA-II, EMS-C, MODE, and MOACS. Except MOHEFT, we curve all the solutions found in 10 runs. Fig. 11 shows some experimental results on the tested five types of real-world workflow with different scale.

From these figures, we can see that the overall performance of MOACS is better than the other five approaches and is significantly superior to ECMSMOO, NSGA-II, and MODE. In small-scale workflows such as Montage 25, MOHEFT, and EMS-C's performance are still close to MOACS. But with the growth of the workflows' scale, MOACS's superiority becomes more and more obvious. Particularly, in those large-scale workflow such as CyberShake 1000, Epigenomics 997, and Inspiral 1000, MOHEFT cannot be completed within an acceptable time while MOACS can generate the well-diversity solutions with lower WEC under the similar WET compare with EMS-C. On the one hand, the new pheromone update rule and the CHS provide good guidance during the search process and help MOACS approach the PF gradually. On the other hand, with the ESS that utilizes the elite knowledge in the global archive to improve the solution quality, MOACS can further approach the global PF. These advantages help MOACS to generate a solution set that has better quality than the other approaches.

Since MOACS is a stochastic approach, further statistical tests are needed to validate its performance. Herein, the HV values in 10 runs are adopted for statistical tests. MOHEFT is not included in statistical tests since it is not a stochastic approach. Both ANOVA and Wilcoxon rank-sum test (also known as Mann–Whitney U test) are used in the statistical tests. The results are shown in Table VII, where the "*A*" column represents the results of ANOVA and the "*W*" column represents the results of Wilcoxon rank-sum test. In the *A* column, if MOACS and the compared approaches are significantly different, the result is "*S*," otherwise the results is "NS." In the *W* column, if MOACS is significantly better than the compared approaches, the result is ">"; if MOACS is significantly worse, the result is "<." The results show that in

TABLE VI
COMPARISONS OF MOACS WITH THE OTHER MULTIOBJECTIVE OPTIMIZATION APPROACHES ON $C(A, B)$ VALUE

Workflow	ECMSMOO		MOHEFT		NSGA-II		EMS-C		MODE	
	C(MOACS,--)	C(--,MOACS)	C(MOACS,--)	C(--,MOACS)	C(MOACS,--)	C(--,MOACS)	C(MOACS,--)	C(--,MOACS)	C(MOACS,--)	C(--,MOACS)
CyberShake 30	**100.00%**	**0.00%**	**100.00%**	29.43%	**100.00%**	**1.42%**	81.19%	67.73%	**100.00%**	**0.00%**
CyberShake 50	**100.00%**	**0.00%**	**100.00%**	36.89%	**100.00%**	**0.00%**	59.84%	83.56%	**100.00%**	**0.00%**
CyberShake 100	**100.00%**	**0.00%**	56.00%	66.26%	**100.00%**	**0.00%**	82.19%	79.75%	**100.00%**	**0.00%**
CyberShake 1000	**100.00%**	**0.00%**	**N/A**	**N/A**	**100.00%**	**0.00%**	84.66%	46.36%	**100.00%**	**0.00%**
Epigenomics 24	**100.00%**	**0.00%**	80.00%	20.00%	88.97%	22.11%	22.66%	100.00%	**100.00%**	**0.00%**
Epigenomics 46	**100.00%**	**0.00%**	**100.00%**	**0.00%**	**91.88%**	21.34%	67.49%	91.30%	**100.00%**	**0.00%**
Epigenomics 100	**100.00%**	**0.00%**	**100.00%**	**0.00%**	**100.00%**	**0.00%**	80.98%	68.12%	**100.00%**	**0.00%**
Epigenomics 997	**100.00%**	**0.00%**	**N/A**	**N/A**	**100.00%**	**0.00%**	**95.51%**	20.59%	**100.00%**	**0.00%**
Inspiral 30	**100.00%**	**0.00%**	**91.30%**	21.85%	**100.00%**	5.54%	**90.33%**	64.31%	**100.00%**	**0.00%**
Inspiral 50	**100.00%**	**0.00%**	**96.30%**	15.57%	**100.00%**	**0.35%**	**98.53%**	28.89%	**100.00%**	**0.00%**
Inspiral 100	**100.00%**	**0.00%**	66.67%	17.47%	**100.00%**	**0.84%**	**99.34%**	10.04%	**100.00%**	**0.00%**
Inspiral 1000	**100.00%**	**0.00%**	**N/A**	**N/A**	**100.00%**	**0.00%**	**99.18%**	**9.40%**	**100.00%**	**0.00%**
Montage 25	**100.00%**	**0.00%**	**100.00%**	30.39%	**100.00%**	**4.95%**	**91.15%**	60.42%	**100.00%**	**0.00%**
Montage 50	**100.00%**	**0.00%**	89.29%	22.14%	**100.00%**	**0.00%**	**100.00%**	19.47%	**100.00%**	**0.00%**
Montage 100	**100.00%**	**0.00%**	69.70%	42.76%	**100.00%**	**3.80%**	**99.56%**	16.28%	**100.00%**	**0.00%**
Montage 1000	**100.00%**	**0.00%**	**N/A**	**N/A**	**100.00%**	**0.00%**	84.82%	72.89%	**100.00%**	**0.00%**
SIPHT 30	**100.00%**	**0.00%**	75.00%	88.55%	**98.25%**	20.26%	67.28%	100.00%	**97.83%**	**0.00%**
SIPHT 60	**100.00%**	**0.00%**	35.29%	89.10%	**100.00%**	**0.00%**	70.11%	69.23%	**100.00%**	**0.00%**
SIPHT 100	**100.00%**	**0.00%**	54.17%	73.95%	**100.00%**	**0.00%**	87.68%	73.49%	**100.00%**	**0.00%**
SIPHT 1000	**100.00%**	**0.00%**	**N/A**	**N/A**	**92.31%**	**1.63%**	68.15%	74.81%	**100.00%**	**0.00%**

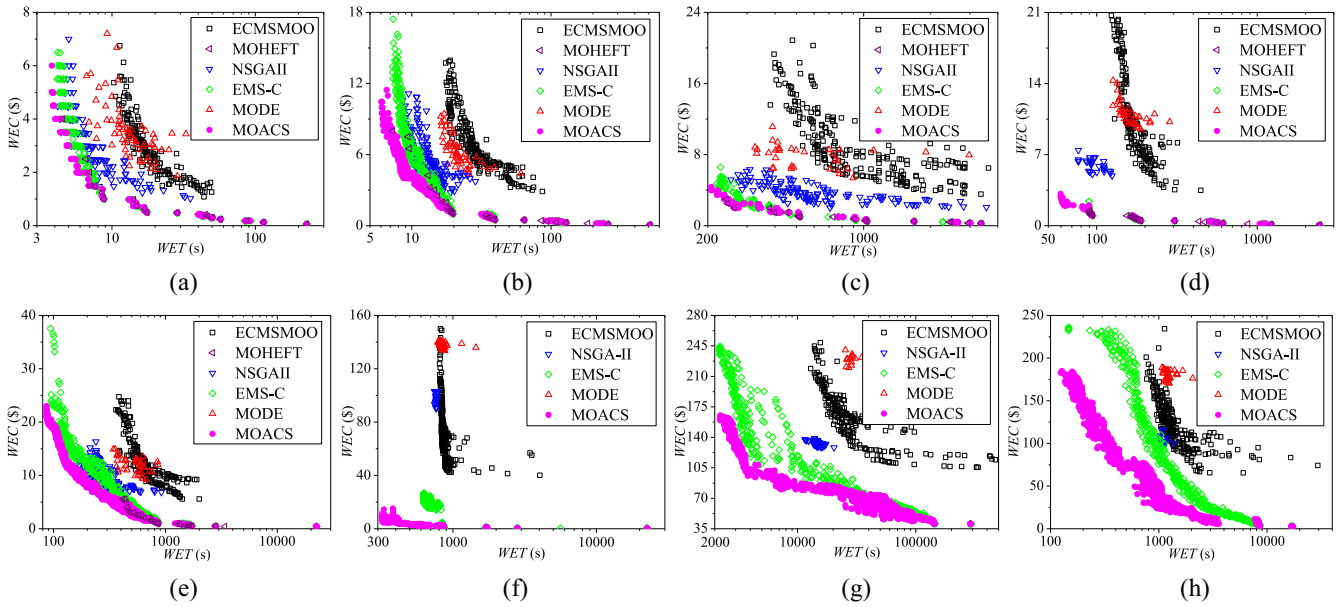

Fig. 11. Performance of ECMSMOO, MOHEFT, NSGA-II, EMS-C, MODE, and MOACS on some tested workflows. (a) Montage 25. (b) Montage 50. (c) SIPHT 60. (d) CyberShake 100. (e) Inspiral 100. (f) CyberShake 1000. (g) Epigenomics 997. (h) Inspiral 1000.

most of the cases, MOACS is significantly different from the compared approaches according to ANOVA, and is significantly better than them according to Wilcoxon rank-sum test. One notice is that in Epigenomics 24, EMS-C is significantly better than MOACS according to Wilcoxon rank-sum test but ANOVA shows that the performance of EMS-C and MOACS are not significantly different.

Efficiency is also an important criterion. Table VIII shows the comparison of MOACS with the other five approaches on the run time. The results are represented by RunTime (other approaches)/RunTime (MOACS). A number greater than 1 shows that MOACS is more efficient. The larger the number is, the more obvious of the advantage. In most of the small-scale workflows, MOACS outperforms the other five approaches. With its high time complexity, the efficiency of MOHEFT drops rapidly with the increase of the workflow scale. The SIPHT workflow has more parallelable tasks, as we can see intuitively in Fig. 10, which results in the large resource pool according to the solution encoding of MOACS (detail in Section III-A). Since MOACS should maintain pheromone and heuristic information for every resource, more time should be spent when dealing with SIPHT workflow. Therefore, the advantage of MOACS's efficiency is weakened in SIPHT workflow and even EMS-C has slightly better efficiency in this case. However, MOACS's efficiency still outperforms EMS-C distinctly in those large-scale workflows. The operators of ECMSMOO and NSGA-II are simple so that with the scale of workflow grows, these two approaches can still complete execution quickly, but the execution results of them are quite unsatisfying according to the previous analysis.

3) Comparison With Constrained Optimization Approaches: Table IX shows the comparison of the WEC

TABLE VII
ANOVA AND WILCOXON RANK-SUM TEST ON THE HV RESULTS OF MOACS AND THE OTHER STOCHASTIC MULTIOBJECTIVE OPTIMIZATION APPROACHES

Workflow	ECMSMOO		NSGA-II		EMS-C		MODE	
	A	W	A	W	A	W	A	W
CyberShake 30	S	>	S	>	S	>	S	>
CyberShake 50	S	>	S	>	S	>	S	>
CyberShake 100	S	>	S	>	S	>	S	>
CyberShake 1000	S	>	S	>	S	>	S	>
Epigenomics 24	S	>	S	>	NS	<	S	>
Epigenomics 46	S	>	S	>	S	>	S	>
Epigenomics 100	S	>	S	>	S	>	S	>
Epigenomics 997	S	>	S	>	S	>	S	>
Inspiral 30	S	>	S	>	S	>	S	>
Inspiral 50	S	>	S	>	S	>	S	>
Inspiral 100	S	>	S	>	S	>	S	>
Inspiral 1000	S	>	S	>	S	>	S	>
Montage 25	S	>	S	>	S	>	S	>
Montage 50	S	>	S	>	S	>	S	>
Montage 100	S	>	S	>	S	>	S	>
Montage 1000	S	>	S	>	S	>	S	>
SIPHT 30	S	>	S	>	S	<	S	>
SIPHT 60	S	>	S	>	S	>	S	>
SIPHT 100	S	>	S	>	S	>	S	>
SIPHT 1000	S	>	S	>	S	>	S	>

TABLE VIII
COMPARISONS OF MOACS WITH THE OTHER MULTIOBJECTIVE OPTIMIZATION APPROACHES ON RUN TIME [RUNTIME (OTHER APPROACHES)/RUNTIME (MOACS)]

Workflow	ECMSMOO	MOHEFT	NSGA-II	EMS-C	MODE
CyberShake 30	**19.142**	**6.658**	**4.325**	**1.352**	**4.766**
CyberShake 50	**4.038**	**7.518**	**1.917**	**1.246**	**2.798**
CyberShake 100	**2.026**	**10.922**	0.819	**2.073**	**3.108**
CyberShake 1000	0.526	**N/A**	0.556	**21.713**	**53.740**
Epigenomics 24	**32.708**	**18.892**	**18.963**	**4.311**	**19.374**
Epigenomics 46	**31.091**	**31.822**	**9.197**	**6.980**	**12.098**
Epigenomics 100	**10.157**	**57.164**	**3.192**	**8.432**	**11.863**
Epigenomics 997	**1.798**	**N/A**	**1.446**	**61.688**	**113.953**
Inspiral 30	**19.218**	**14.714**	**9.599**	**2.953**	**10.045**
Inspiral 50	**9.986**	**18.738**	**4.317**	**2.913**	**6.047**
Inspiral 100	**4.573**	**31.332**	**1.760**	**4.846**	**6.369**
Inspiral 1000	0.976	**N/A**	0.906	**33.674**	**62.195**
Montage 25	**15.594**	**7.345**	**6.551**	**1.706**	**6.716**
Montage 50	**3.365**	**11.885**	**1.517**	**1.058**	**2.174**
Montage 100	**1.427**	**16.261**	0.582	**1.296**	**2.092**
Montage 1000	0.241	**N/A**	0.215	**9.356**	**17.782**
SIPHT 30	**6.901**	**15.170**	**2.282**	0.884	**2.434**
SIPHT 60	**2.029**	**28.696**	0.893	0.847	**1.530**
SIPHT 100	0.932	**43.779**	0.408	0.933	**1.501**
SIPHT 1000	0.163	**N/A**	0.149	**6.458**	**12.328**

of the solutions found by PSO, DOGA, and MOACS under the same deadline constraint. For PSO and DOGA, we use the average WEC in 10 runs for comparison. For MOACS, we use the average WEC of those solutions that have the minimal WEC under the deadline constraint in each run for comparison. The value of WEC (other approaches)/WEC (MOACS) can represent the comparison of PSO, DOGA, and MOACS directly. A number greater than 1 represents that MOACS's performance is better. The larger the number is, more obvious of the advantage. The table shows that MOACS can find a solution with smaller WEC. With the workflow scale grows, the superiority of MOACS is more distinct.

For constrained optimization approaches, we set a series of deadlines, beginning with a large enough deadline, in descending order until the approaches cannot find a feasible solution. In this way, PSO and DOGA can also generate a series of feasible solutions. We curve the solutions found by PSO, DOGA, and MOACS in 10 independent runs and some of the experimental results are shown in Fig. 12.

TABLE IX
COMPARISONS OF WEC OF THE SOLUTIONS FOUND BY PSO, DOGA, AND MOACS UNDER THE SAME DEADLINE CONSTRAINT [WEC (OTHER APPROACHES)/WEC (MOACS)]

Workflow	DEADLINE	PSO/MOACS	DOGA/MOACS
CyberShake 30	100	**2.928**	**8.616**
CyberShake 50	200	**5.208**	**16.012**
CyberShake 100	300	**13.676**	**36.560**
CyberShake 1000	1000	**116.292**	**183.502**
Epigenomics 24	1000	**1.745**	**4.069**
Epigenomics 46	4000	**1.839**	**4.408**
Epigenomics 100	7000	**4.459**	**15.066**
Epigenomics 997	50000	**4.388**	**5.646**
Inspiral 30	300	**2.320**	**6.732**
Inspiral 50	400	**3.184**	**9.901**
Inspiral 100	800	**6.596**	**16.174**
Inspiral 1000	2000	**12.289**	**16.737**
Montage 25	40	**2.543**	**7.233**
Montage 50	80	**5.688**	**18.288**
Montage 100	100	**17.204**	**41.744**
Montage 1000	700	**212.972**	**276.944**
SIPHT 30	400	**2.930**	**8.098**
SIPHT 60	600	**7.412**	**17.472**
SIPHT 100	1000	**14.684**	**37.503**
SIPHT 1000	3000	**5.169**	**6.591**

In these figures, we can see that the performance of MOACS is better than both PSO and DOGA. First of all, comparing the solutions with the similar WET, MOACS can generate solutions with lower WEC. Second, with the growing scale of the workflow, MOACS's advantage becomes more and more obvious. Particularly, in the large-scale workflows such as CyberShake 1000 and Inspiral 1000, PSO and DOGA do not have well global search ability to deal with the large search space, resulting in very poor performance. While the new pheromone update rule and the CHS in MOACS maintain good global search ability, and the ESS helps further approach the global PF. More interesting, the solutions found by DOGA and PSO in different runs often have different distributions, while the solutions found by MOACS distribute more stable along the PF. Therefore, MOACS may be more preferred due to its stable search ability.

D. Parameter Study

In MOACS, the settings of β, q_0, ρ, and ε are the typical scheme in ACS so we focus on the other three parameters, *ess_num*, *ess_rate*, and *sr*. Note that when testing a parameter, the other parameters of MOACS are set according to Section IV-B.

Fig. 13 shows the parameter study on *ess_num*, *ess_rate*, and *sr*. For each figure, the *x*-axis are several parameter settings and the *y*-axis are the average HV value in 10 runs corresponding to these settings. The dashed lines mark the selected settings. From Fig. 13(a), we can see that our selected setting "30" for *ess_num* is the best among the other settings. In Fig. 13(b), the setting "0" for *ess_rate* represents that MOACS only employs the large-scope ESS while the setting "1" represents that only the small-scope ESS is employed. Our selected settings "0.2" has the best performance in most of the workflows, particularly in SIPHT 100 and Inspiral 1000. In Fig. 13(c), the setting 0 for *sr* represents that the colony greedily selects the solution with the smallest value of its optimization objective for the pheromone global update.

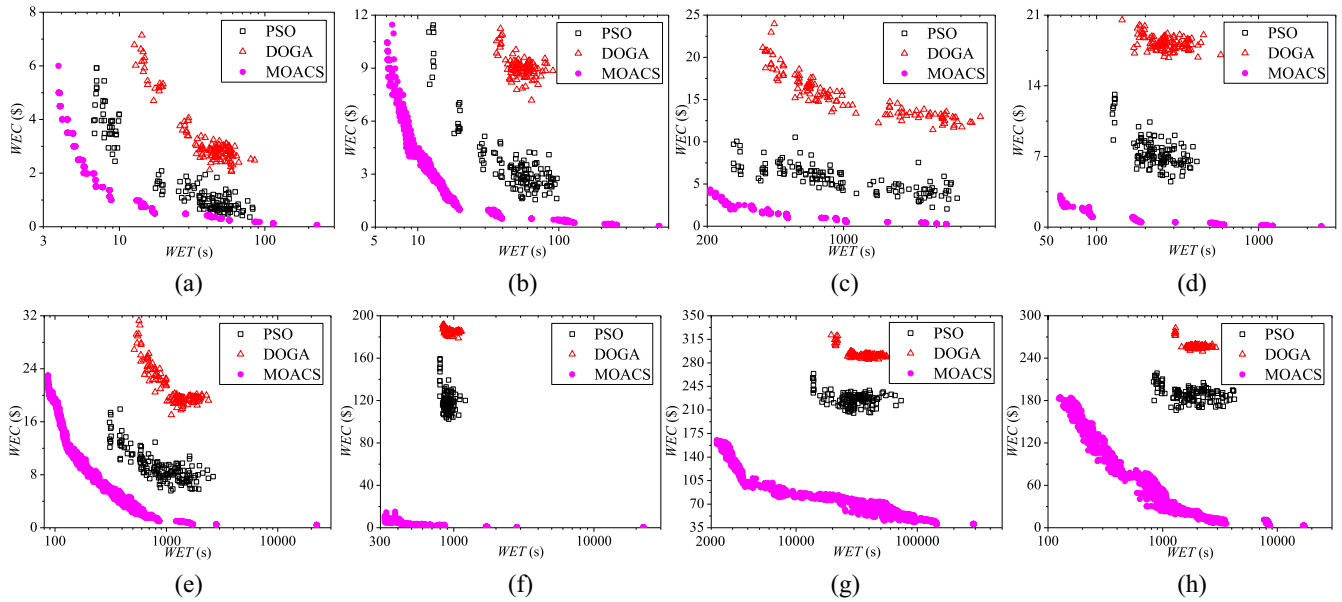

Fig. 12. Performance of PSO, DOGA, and MOACS on some tested workflows. (a) Montage 25. (b) Montage 50. (c) SIPHT 60. (d) CyberShake 100. (e) Inspiral 100. (f) CyberShake 1000. (g) Epigenomics 997. (h) Inspiral 1000.

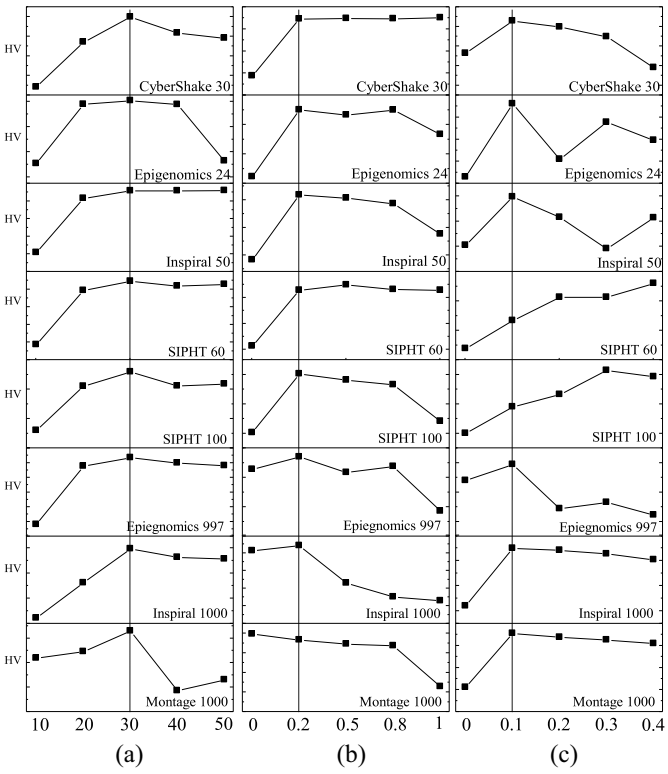

Fig. 13. Parameter study on (a) *ess_num*, (b) *ess_rate*, and (c) *sr*.

We can see that in SIPHT 60 and SIPHT 100, our selected setting "0.1" for *sr* is worse than the other settings like "0.3" and "0.4." However, our selected setting still performs better in the other workflows, particularly in Epigenomics 997.

V. Conclusion

In this paper, we propose a novel MOACS approach for cloud workflow scheduling, with optimization objectives of WET and WEC. Two ant colonies are adopted to optimize execution time and execution cost, respectively. MOACS well considers the features of cloud computing since it adopts a heterogeneous resource pool and adopts "maximal parallel tasks" to simulate the elasticity of cloud computing, which has high capacity to be extended to the actual cloud platform. A new pheromone update rule is designed based on a set of nondominated solutions from a global archive to maintain the diversity and guarantee the search efficiency, which can guide each colony to search its optimization objective sufficiently. In order to avoid a colony focusing only on its own optimization objective, the CHS is proposed to ensure the quality of the other objective. The pheromone update rule and the CHS help the algorithm approach the PF gradually. Moreover, the ESS is performed to improve the solution quality of the global archive so as to help further approach the global PF.

Our experiments are simulated based on the data of Amazon EC2 cloud platform and five types of real-world workflows from different scientific areas. The experimental results show that the performance of MOACS in solving cloud workflow scheduling is better than not only the multiobjective optimization approaches (ECMSMOO, MOHEFT, NSGA-II, EMS-C, and MODE) but also those constrained optimization approaches (PSO and DOGA). First of all, MOACS can generate a solution with similar WET but lower WEC than the other approaches. Second, with the growing scale of the workflows, MOACS's advantage is more distinct. Third, MOACS has better global search ability, particularly when dealing with the large-scale workflows, as it can generate more nondominated solutions that are also widely distributed in the solution space. In the future work, other cloud environments or even multiclouds environments should be adopted to test the performance of MOACS.

REFERENCES

[1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, document 800–145, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, 2011.

[2] K. Mershad, H. Artail, M. A. R. Saghir, H. Hajj, and M. Awad, "A study of the performance of a cloud data center server," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 590–603, Oct./Dec. 2017.

[3] M. Armbrust *et al.*, "A view of cloud computing," *Comm. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] H. Yuan *et al.*, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[5] T. Atmaca, T. Begin, A. Brandwajn, and H. Castel-Taleb, "Performance evaluation of cloud computing centers with general arrivals and service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2341–2348, Aug. 2016.

[6] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 171–209, 2014

[7] I. Sadooghi *et al.*, "Understanding the performance and potential of cloud computing for scientific applications," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 358–371, Apr./Jun. 2017.

[8] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[9] L. Wang, M. Liu, and M. Q. H. Meng, "A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 473–484, Feb. 2017.

[10] G.-P. Liu, "Predictive control of networked multiagent systems via cloud computing," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 1852–1859, Aug. 2017.

[11] 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.

[12] S.-K. Chou, M.-K. Jiau, and S.-C. Huang, "Stochastic set-based particle swarm optimization based on local exploration for solving the Carpool service problem," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1771–1783, Aug. 2016.

[13] Y. Zhou, S. Kwong, H. Guo, X. Zhang, and Q. Zhang, "A two-phase evolutionary approach for compressive sensing reconstruction," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2651–2663, Sep. 2017.

[14] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Inf. Sci.*, vol. 293, no. 1, pp. 370–382, 2015.

[15] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2017.2769108.

[16] X.-F. Liu *et al.*, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[17] X.-F. Liu, Z.-H. Zhan, and J. Zhang, "An energy aware unified ant colony system for dynamic virtual machine placement in cloud computing," *Energies*, vol. 10, no. 5, pp. 1–15, 2017.

[18] R. Garg and A. K. Singh, "Multi-objective workflow grid scheduling based on discrete particle swarm optimization," in *Proc. Int. Conf. Swarm Evol. Memetic Comput.*, 2011, pp. 183–190.

[19] A. K. M. K. A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency Comput. Pract. Exp.*, vol. 21, no. 13, pp. 1742–1756, 2009.

[20] Z. H. Zhan *et al.*, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surveys*, vol. 47, no. 4, p. 63, 2015.

[21] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. Grid Comput. Environ. Workshop*, Austin, TX, USA, 2008, pp. 1–10.

[22] N. R. Herbst, S. Kounev, and R. H. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *Proc. Int. Conf. Auton. Comput.*, San Jose, CA, USA, 2013, pp. 23–27.

[23] S. Raghavan, P. Sarwesh, C. Marimuthu, and K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud," in *Proc. Int. Conf. Electron. Design Comput. Netw. Autom. Verification*, 2015, pp. 139–144.

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

[25] 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. IEEE Int. Conf. Comput. Intell. Security*, Perth, WA, Australia, 2010, pp. 400–407.

[26] B. Lin *et al.*, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 581–594, Sep. 2016.

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

[28] Z.-G. Chen, K.-J. Du, Z.-H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2015, pp. 708–714.

[29] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Comput.*, vol. 17, no. 2, pp. 169–189, 2014.

[30] 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.

[31] 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.

[32] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[33] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[34] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[35] T. Liao, K. Socha, M. A. M. de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 503–518, Aug. 2014.

[36] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, Jul. 2017.

[37] Z.-G. Chen *et al.*, "Deadline constrained cloud computing resources scheduling through an ant colony system approach," in *Proc. Int. Conf. Cloud Comput. Res. Innovat.*, 2015, pp. 112–119.

[38] D. Angus and C. Woodward, "Multiple objective ant colony optimization," *Swarm Intell.*, vol. 3, no. 1, pp. 69–85, 2009.

[39] M. Lopez-Ibanez and T. Stützle, "The automatic design of multiobjective ant colony optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 16, no. 6, pp. 861–875, Dec. 2012.

[40] M. López-Ibáñez and T. Stützle, "An experimental analysis of design choices of multi-objective ant colony optimization algorithms," *Swarm Intell.*, vol. 6, no.3. pp. 207–232, 2012.

[41] Z.-H. Zhan *et al.*, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[42] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 2–18, Jan./Mar. 2018.

[43] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[44] S. Ostermann *et al.*, "A performance analysis of EC2 cloud computing services for scientific computing," in *Proc. Int. Conf. Cloud Comput.*, 2010, pp. 115–131.

[45] S. Bharathi *et al.*, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large Scale Sci.*, Austin, TX, USA, 2008, pp. 1–10.

[46] G. Juve *et al.*, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, 2013.

[47] G. Yao, Y. Ding, Y. Jin, and K. Hao, "Endocrine-based coevolutionary multi-swarm for multi-objective workflow scheduling in a cloud system," *Soft Comput.*, vol. 21, no. 15, pp. 4309–4322, 2017.

[48] 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.

Zong-Gan Chen (S'17) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in computer science and technology with the South China University of Technology, Guangzhou.

His current research interests include ant colony optimization, differential evolution, and their applications in real-world optimization problems.

Zhi-Hui Zhan (M'13) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China, in 2014, the Pearl River New Star in Science and Technology in 2015, the Youth Talent in Science and Technology Innovation of Guangdong Province in 2016, the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017, and the China Computer Federation Outstanding Dissertation Award and the IEEE CIS Outstanding Dissertation Award for his doctoral dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science.

Ying Lin (M'12) received the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Assistant Professor with the Department of Psychology, Sun Yat-sen University and also a Research Fellow with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou. Her current research interests include computational intelligence and its applications in network analysis, cognitive diagnosis, and cloud computing.

Yue-Jiao Gong (M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

From 2015 to 2016, she was a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau, China. She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. She is also a Young Pearl River Scholar. Her current research interests include evolutionary computation and machine learning methods, as well as their applications to intelligent transportation and smart city.

Tian-Long Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987 and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Bentley, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Murdoch, WA, Australia. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Feng Zhao received the Ph.D. degree in communication and information system from Shandong University, Jinan, China, in 2007.

He is currently a Full Professor with the School of Electronics and Communication Engineering, Yulin Normal University, Yulin, China. His current research interests include cognitive radio networks, MIMO wireless communications, cooperative communications, smart antenna techniques, and cloud computing.

Hua-Qiang Yuan received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1996.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include computational intelligence and cyberspace security.

Xiaofeng Chen (SM'16) received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 2003.

He is currently a Professor with Xidian University. He has published over 100 research papers in refereed international conferences and journals. He has over 4000 Google Scholar Citations. His current research interests include applied cryptography and cloud computing security.

Dr. Chen is in the Editorial Board of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, *Security and Communication Networks*, and *Computing and Informatics*. He has served as the program/general chair or program committee member in over 30 international conferences.

Qing Li (SM'07) received the B.Eng. degree in computer science from Hunan University, Changsha, China, and the M.Sc. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, CA, USA.

He is the Founding Director of the Multimedia Software Engineering Research Centre, and concurrently a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include dynamic object modeling, multimedia and mobile information retrieval and management, distributed databases and data warehousing/mining, and workflow management and Web services.

Jun Zhang (F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Changjiang Chair Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has published over 100 technical papers in the above areas.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.