IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2020, accepted February 6, 2020, date of publication February 10, 2020, date of current version February 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972963

Makespan-Driven Workflow Scheduling in Clouds Using Immune-Based PSO Algorithm

PENGWEI WANG, YINGHUI LEI, PROMISE RICARDO AGBEDANU, AND ZHAOHUI ZHANG

School of Compute Science and Technology, Donghua University, Shanghai 201620, China

Corresponding author: Pengwei Wang (wangpengwei@dhu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61602109, in part by the Shanghai Science and Technology Innovation Action Plan under Grant 19511101802, in part by the Natural Science Foundation of Shanghai under Grant 19ZR1401900, in part by the Fundamental Research Funds for the Central Universities, and in part by the DHU Distinguished Young Professor Program under Grant LZB2019003.

ABSTRACT Cloud Computing is becoming more and more popular for solving problems that need high concurrency and a lot of resources. Many traditional areas of research choose to solve their problems through the cloud, and workflow scheduling is one of them. Cloud computing brings many benefits, meanwhile, due to the almost ''infinite'' amount of resources for users, it also brings new challenges for scheduling and optimization, in which cost and makespan are the most concerned issues for workflow scheduling. Users want to obtain a low cost and fast makespan solution. This paper focuses on how to find an optimized solution to achieve better cost-makespan at the same time under the constraint of deadline. In order to solve this problem, an immune particle swarm optimization algorithm (IMPSO) is proposed, which effectively improves the quality and speed of the optimization. The proposed IMPSO overcomes the problem of slow convergence of PSO, which is easy to fall into local optimization. Experiments show the efficiency and effectiveness of the proposed approach.

INDEX TERMS Cloud computing, workflow scheduling, immune mechanism, particle swarm algorithm.

I. INTRODUCTION

Workflow is a series of tasks ordered as some form of prioritization to achieve a specific aim, which is a branch of discrete event systems (DES). Nowadays, science and business applications consist of thousands of tasks, even more. As usual, for convenient research, they are modeled as workflow model via direct acyclic graphs (DAG) or Petri nets [1]–[4]. These complex workflow applications are deployed in a distributed computing environment in order to execute them in a reasonable amount of time [5]. But there is a question of reality that, building and maintaining an infrastructure, such as high-performance clusters, grid systems or private clouds, by the application owners is not only very expensive, but also cannot deal with dynamic demands flexibly. When demand is greater than the existing facilities, the infrastructure has to expand, and if there is a significant reduction in demand for a long period, a number of resources are wasted. Therefore, cloud computing; a shared-resource mode that provides services as infrastructure, where users can deploy their workflows dynamically. Moreover, users can pay and use the cloud resources on demand.

In recent years, cloud computing has developed extremely rapidly. As a mature business model, cloud computing is provided by many providers in the worldwide market which forms a benign competition. Thus, users can enjoy high quality service at low prices. Normally, cloud providers divide the world into several regions, and builds datacenters in each region. Each datacenter provides numerous instance types, which are distinguished from each other by some parameters, like CPU, memory, bandwidth, etc. The resources a user can obtain from the market can be seen as ''infinite'', due to the order of magnitude gap between user and instances offered. It means that it is difficult for a user to judge between using which type of instances and how many instances can accomplish the workflow at a lower price and lower makespan. Therefore, workflow scheduling in cloud environment is very significant and meaningful.

Workflow scheduling in cloud environment aims to allocate each task in the workflow to a certain resource for

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

execution in order to meet some performance criterion [5]. Due to numerous instances offered to users, it is almost impossible to traversal all resources to find the best solution in polynomial time. Moreover, users often have requirements on more than one performance criterions. For example, the solution does not only have to been the lowest cost, but also the fastest makespan, which makes the scheduling problem more complicated. Workflow scheduling is a well-known NP-hard problem, which has caused extensive research, especially scheduling in clouds. Cloud instance selection is also another issue, there are also some researches about them, such as [6]–[8].

Deploying workflow in clouds is different from doing so on traditional distributed systems. It means that cost is dynamic, which is decided by the instances a user gets. Finding a better instances solution is a deserved research. In order to find an optimized solution to achieve better cost-makespan at the same time under the constraint of deadline, we propose an immune-based particle swarm optimization (IMPSO) algorithm. Making use of the characteristics of immune mechanism as immunological memory, affinity evaluation between antibody and antigen, antibody concentration, and degree of excitability and so on. Experimental evaluation proves that the performance of IMPSO is competitive and the convergence speed is obviously faster than that of the contrast algorithm. For makespan and cost, IMPSO has a certain degree of improvement.

The reminder of this paper is organized as follows: Section II reviews the related work. Section III describes the scheduling model and problem formulation, while Section IV talks about the optimization approach. Experimental results are shown in Section V. Finally, Section VI concludes this work and points to future work.

II. RELATED WORK

Workflow scheduling has been studied for several decades. Researchers put forward a wide variety of ways to get the optimized deployment solution. There are some reviews that systematically describe workflow scheduling problems from different angles. Rodriguez and Buyya [9] review scientific workflow scheduling studies on IaaS environment, and classifies cloud resource model according to different cloud characteristics and services. Smanchat and Viriyapant [10] classify the existing research according to scheduling process, task and resource which complements the workflow classification of grid computing. Kaur *et al.* [11] discuss the heuristic, metaheuristic and hybrid methods about workflow scheduling. Reference [12] focus on using intelligent optimization algorithm to ensure security and obtain a good scheduling scheme at the same time.

A workflow is a collection of dependent tasks that perform specific functions. The dependency of a workflow deployed on a computer is usually represented by the transmission of data and determines the order in which tasks are executed. Tasks in workflow have different requirements for resources. Some want more memory, and some prefer more I/O resources. When executing workflow, it is important not only to meet the completion time of the whole workflow, but also to meet the resource requirements of each task, which requires that individuals or organizations need enough budget and space to build the corresponding infrastructure. In addition, when the workflow requirements change or need to execute a new workflow with different requirements, the corresponding infrastructure should also be adjusted. However, not every individual or organization has the ability to build and adjust infrastructure, so researchers are eager to find new computing models to serve the workflow. Cloud computing is the most typical representative.

Compared with traditional techniques, in the cloud environment, not only makespan is considered, other objects such as cost, load balance, safety, energy, reliability are also critical. Aiming to different objects, a great quantity of methods are proposed to get a better solution for making one or more objects best. Makespan and Cost are the most considered parameter.

The existing studies can be generally divided into three types: based on Pareto solution, based on weight and based on given constraints [13]. In practical applications, users often consider more than one goal. such as cost, completion time, safety and reliability. there are trade-offs between these objectives, and this is a NP-hard problem that cannot be solved in polynomial time.

A. MAKESPAN-DRIVEN WORKFLOW SCHEDULING

In recent researches, cost and makespan are still the most important concerns. Makespan is the time to perform a full workflow. It is also referred to as a scheduling length. As the focus of the work flow scheduling problem in the grid calculation and the traditional calculation mode. Many well-known scheduling algorithms have been applied, such as Myopic [14], Min-min [15], Max-min [16], GRASP [16] etc. Many methohds solving workflow scheduling are based on heuristic, and can be classified as List schedule [17], clustered heuristic schedule [18] and task replication heuristic schedule [19]. Among them, the earliest time first (ETF) algorithm [20] and the heterogeneous earliest completion time (HEFT) algorithm have become the most commonly used ideas in makespan modeling. In addition, a heuristic idea is called critical path (CP) [21], which determines the longest of all execution paths in workflow (critical path) and gives the highest priority to them in order to minimize the completion time of the whole workflow. CP has been widely used to deal with the scheduling of interdependent tasks in multiprocessor systems. Using the idea of fill control, a heuristic of Firefly guidelines based on the travel behavior of fireflies is established to distribute the load equally, thus reducing the overall completion time [22]. To schedule big data workflow, Mohan *et.al.* [23] define a big data workflow budget constraint called BARENTS. This constraint supports the scheduling of high-performance workflows in heterogeneous cloud computing environments, which can minimize workflow completion times under budget constraints.

The above-described scheduling is mostly static, that is, considering only the current network state, generating a good scheduling plan, and the lack of consideration of resource availability changes. In cloud computing, however, resources are often dynamic, and the actual execution time of the workflow is often longer than the time predicted in static mode. There are some dynamic scheduling solutions, Jiang *et.al.* [24] propose a method called dynamic Earliest-Finish-Time (DEFT) in IaaS cloud to improve both makespan and robustness. It contains a set of list scheduling loops. In each cycle, rank unplanned tasks, select an optimal virtual machine (VM), and estimate the minimum earliest completion time for each task.

Makespan is a basic attribute of workflow. Whether the optimization goal is cost, resource utilization or energy consumption, it should be optimized under the premise of meeting the completion time.

B. COST-DRIVEN WORKFLOW SCHEDULING

As one of the most concerned objects, researches pay much attention to optimal cost. In most cost optimization decisions, cost optimization is usually considered with deadline constraints. Otherwise, in order to make the cost as low as possible, it is necessary to select the cheapest instance type to execute the workflow, but this soluton may increase deadline violation significantly. For solving this, the characteristics of cloud resources need to be carefully considered.

In order to solve the problem of cost optimization, the characteristics of cloud resources need to be carefully considered. A classical heuristic solution is a static algorithm IC-PCP based on partial critical path (PCP) proposed by Abrishami *et al.* [25] and IC-PCPD2, scheduling based on workflow partial critical path. This method takes into account the heterogeneity of the cloud, that is, the pricing model of on-demand payment and time interval, and tries to deploy all the tasks on some critical paths as an example on the premise of ensuring the deadline. However, these methods do not have the global optimization technique to approximate the optimal solution. To some extent, the use of the whole workflow structure and characteristics is lost by using the method of task classification. In order to improve this problem, Rodriguez and Buyya [26] develop a combined resource allocation and scheduling strategy for IaaS environment, which considers the dynamic configuration of unlimited cloud resources and the performance change of instances. The meta-heuristic optimization algorithm PSO is used to solve the problem of minimizing the cost under the premise of satisfying the deadline of users. For considering the requirement of completion time for workflows, a two-stage algorithm is proposed to optimize the cost under delay-based constraints [27]. In order to improve the quality of service (QoS) and effectively reduce the operating cost of cloud workflow, a scheduling model oriented to QoS and cost is established [28]. With tenant lease and virtual machine instance load as constraints, the savings of cloud service cost and cloud resource cost are realized by population iteration evolution in two links of genetic reorganization and mutation.

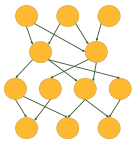


FIGURE 1. A sample DAG with 12 tasks.

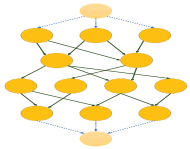


FIGURE 2. A simple DAG with dummy tasks.

III. SCHEDULING MODEL AND PROBLEM FORMULATION

In this section, we will talk about our model of workflow scheduling in cloud environment and give the formulation of the problem we consider.

A. SCHEDULING MODEL

For a convenient description, workflow is often modeled as directed acyclic graph (DAG). It assumes that the scheduler manages workflow execution in a sequential and independent manner. In this way, the scheduling algorithm can optimize the cost of a single a user and a single DAG to meet the requirements of quality of service (QoS). In general, a workflow can be represented as a weighted DAG graph $G = (T, E)$, where T is a set of n tasks$\{t_1, t_2, t_3, \ldots, t_n\}$, each task is individual and has a certain amount of workload len(t_i). E is a set of precedence dependencies among tasks. $E = \{e_{i,j} | i, j = 0, 1, 2, \ldots, N\}$, a precedence dependencies $e_{i,j}$ represents precedence constraint between task t_i and task t_j. It means that t_j can start executing just when t_i finishes, if there are data communication between two tasks, t_j must wait for data transfer to finish. Moreover, t_i is called t_j's *parent* task or *predecessor* task, and t_j is called t_i's *child* task or *subsequent* task. We use $pred(t_i)$ to represent the set of parent tasks of task t_i. For each task, it can have zero, one or more parent tasks and child tasks. In DAG graph, we call those tasks without parent task entry task, and those without child task exit task. There is $pred(t_{entry}) = \emptyset$. In order to facilitate the design of scheduling method, two dummy task t_{entry} and t_{exit} with zero execution time and zero data transfer add to the beginning and the end of workflow, respectively. Like Fig. 1 and Fig. 2.

In cloud environment model, cloud provider P provides 'infinite' instances $I = I_s$ with different processing capabilities and different rent costs. Price model uses fixed price like on-demand instance of Amazon EC2, $price(I_s)$ indicates that the cost of instance t in a time interval. Users are charged on the basis of the number of time intervals that they used and any partial utilization is regarded as a full time interval

(e.g., 0.1 second is rounded to 1 second). We use $capacity_s$ to represent instance's processing capability.

When task t_i is allocated to instance, the execution time can be calculated via,

$$ET\ (t_i, I) = \frac{len\ (t_i)}{capacity_I} \tag{1}$$

And the execution cost of task t_i is:

$$EC(t_i, I) = ET(t_i, I) \times price(I_s) \tag{2}$$

Moreover, if taks t_j is the parent of task t_i, there are $data_{j,i}$ from parent to child task. We assume that all instances of the provider offered are in the same region. Thus, the bandwidth bw of all instances is roughly equal and if tasks in the same instance, transfer time is free. formula 3 indicates the transfer time between task t_j and task t_i.

$$TT\left(e_{j,i}\right) = \begin{cases} \dfrac{data_{j,i}}{bandwidth}, & if\ i \neq j \\ 0, & otherwise \end{cases} \tag{3}$$

For task t_i, the cost of all data send to it is as formula 4, in which $trans$ is the unit transmission price.

$$TC(t_i) = \sum_{j=1}^{pred(t_i)} data_{j,i} \times trans \tag{4}$$

Total cost of this workflow can be defined as following:

$$COST = \sum_{i=1}^{N} (EC(t_i, I) + TC(t_i)) \tag{5}$$

Let $ST(t_i)$ be the start time and $FT(t_i)$ be the end time of task t_i, which can be calculated by:

$$ST(t_i) = \begin{cases} 0 & pred(t_i) = \emptyset \\ \max\limits_{t_j \in pred(t_i)} \{ST(t_j) & \\ + ET(t_j, I) + TT(e_{j,i})\} & pred(t_i) \neq \emptyset \end{cases} \tag{6}$$

$$FT(t_i) = ST(t_i) + ET(t_i, I) \tag{7}$$

For workflow, the finish time of exit task is the makespan of whole workflow, which is captured formula 8. Besides, all tasks that make up the maximum end time of the exit task constitute the critical path (CP).

$$makespan = FT(t_{exit}) \tag{8}$$

B. PROBLEM FORMULATION

When scheduling workflow in cloud environment, there are two issues considered, one is resource provisioning. It means that the scheduling solution need to decide how many instances to rent, their types, their startup and finish time. Another is actual scheduling or task-instance mapping. These issue focuses on mapping each task onto the most suitable resource. In general, workflow scheduling usually considers the combination of these two issues.

A schedule can be denoted as $\langle I, A \rangle$. $I = \{I_1, I_2, I_3, \ldots I_{|I|}\}$ is a set of used resource and each I_s is associated with a instance type, the lease start $time(LST)$ and the lease end $time(LFT)$. A is the allocation of task to instance, each allocation $A = \langle t_i, s_j, ST(t_i), FT(t_i) \rangle$ represents task t_i is allocated to resource s_j, start at $ST(t_i)$ and end at $FT(t_i)$.

As for the total cost and makespan of a workflow, they can be calculated by formula 5 and 8.

Let D be the user-defined deadline. The deadline-constrained cost and makespan optimization problem for workflow in clouds is formulated as:

$$\begin{aligned} &\min .\ COST \\ &\quad makespan \\ &s.t.\ makespan \leq D \end{aligned} \tag{9}$$

Formula 9 is the main goal of this paper, and we need to find a solution with better cost and makespan. However, due to the time requirement of workflow, the makespan cannot be set to an unlimited existence, so the deadline is set to limit it.

IV. AN IMMUNE-BASED PARTICLE SWARM OPTIMIZATION IN CLOUD ENVIRONMENT(IMPSO)

In this section, we propose an immune-based particle swarm optimization algorithm for workflow scheduling in clouds.

A. PARTICLE SWARM ALGORITHM AND IMMUNE MECHANISM

Particle swarm algorithm (PSO) is an evolutionary computing technique based on fauna behavior. Since it was developed in 1995, it has been widely studied and used. This algorithm is a stochastic optimization technique in which particles represent individuals who can move in the defined problem space and represent candidate solutions to the optimization problem. At a given point in time, the motion of particle is defined by their velocity and represented as a vector. Therefore, it has amplitude and direction. PSO adopts constant learning factor and inertia weight. Formula 10 and 11 show the speed and position of a particle. V_i^k and P_i^k are the speed and position of a particle i in k time, respectively, both of them are vectors. P_i^{pbest} and p_g^{gbest} are the best position of the i-th particle and the global, respectively.

$$V_i^k = \omega V_i^{i-1} + c_1 r_1 (P_i^{pbest} - P_i^{k-1}) + c_2 r_2 (P_g^{gbest} - P_i^{k-1}) \tag{10}$$

$$\mathrm{P}_i^k = P_i^{k-1} + V_i^k \tag{11}$$

Biological immune system is a complex system composed of organs, cells and molecules. It is a functional system in which the body can specifically recognize ''non-self'' and ''oneself'' stimuli, respond accurately to them, and retain memory responses. In short, when foreign objects enter the human body in the form of antigens, the immune system produces antibodies that solve their problems and has the function of memory and self-regulation.

According to the characteristics of biological immune system, artificial immune system is constructed as the same self-regulatory mechanism. Affinity is used to describe the degree of similarity between antibody and antigen, or antibody and

antibody. When solving a multi-objective optimization problem, the problem to be solved is an antigen and candidate solutions are antibodies. The affinity between antibody and antigen reflects the similarity between candidate solution and optimal solution, that is, the satisfaction of candidate solution to constraint conditions and objective function. The affinity between antibodies and antibodies reflects the similarity between different candidate solutions and the diversity of candidate solutions. Antibodies with similarity in a certain range can be quantified as the concentration of an antibody. By adjusting the antibody concentration, we can avoid falling into local optimization in the process of optimization. By screening out antibodies with higher affinity, the process of optimization can be accelerated. Immune memory is that the immune system can retain some of the antibodies that deal with the antigen better as memory cells. When the same antigen invades again, the corresponding memory cells are activated and a large number of antibodies are produced.

The optimization process of PSO is divided into five main stages, first initializing the population and giving each particle a randomly generated speed and position. Next, the suffciency of each particle is calculated, and the best one is identified as the global optimal particle. According to the current position and velocity, the third stage is calculating next velocity and condition according to formula 10 and 11. Finally, update the global optimal particle. It is then determined whether it meets the iterative stop condition. If not satisfied, continue the iteration. In the process of PSO, if one particle finds a current optimal position, the other particles will move closer to it quickly. The aggregation phenomenon will occur, which will lead to the decrease of population diversity. If the current optimal position is the local best, then the particle swarm optimization is difficult to re-search in the solution space, and the algorithm falls into the local optimal. The artificial immune mechanism has no unified process description at present. In this paper, an immune mechanism is introduced in particle swarm optimization. The algorithm flow is shown in Fig. 3. In this algorithm, particles are considered as antibodies. After initializing antibody population and obtaining the initial global optimal solution, each antibody will have an affinity and concentration. Then, an incentive value is obtained to evaluate the excellent degree of this antibody. Next there are a series of operations such as antibody clone, cross and mutation, memory cell formation, replacement of the worst solutions and so on. In the final stage, update the population and judge whether the algorithm meets the end condition. If it meets the requirements, output the results, or the iteration continues.

B. PROPOSED METHOD

The procedure of IMPSO algorithm is shown as algorithm 1. The detail definitions of some operations are as following.

1) ENCODED MODE

When the workflow scheduling problem in the cloud computing environment is modeled as the solution model of

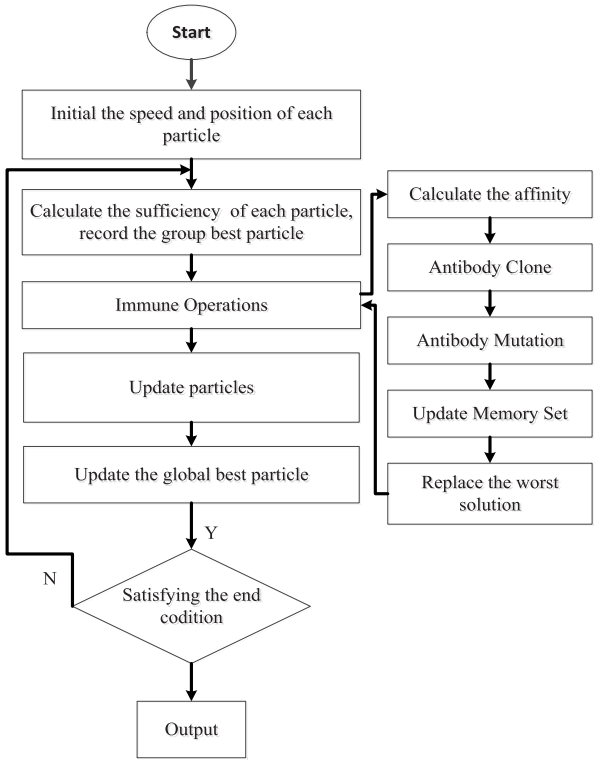


FIGURE 3. The process of IMPSO.

the IMPSO approach, there are two questions. One is what defines the coding mode of this problem. The other is how to evaluate the excellent degree of the particle, that is, the fitness function.

For this proposed approach, a particle represents a scheduling solution. Therefore, the dimension of a particle is equal to the number of tasks in the given workflow. Given a workflow $G = \{T, E\}$, assume that it contains N tasks, then, the particles in this algorithm is an N-dimension particle. The size of the particle will be used in the coordinate system that determines its position in search space. Due to the infinite instances in cloud computing, it is impossible to let the coordinates of particles take arbitrary values in the real range, which is not conducive to fast search. Thus, the range of particle movements is determined by available instance types. Assuming there are M kinds of instance types, the maximum value of particle position is that M. Based on this, the integer part of the coordinate value in the particle position represents the selected instance type.

As for fitness function, it is used to determine the effectiveness of the candidate solution, it is necessary to reflect the objectives of the scheduling problem. That is, cost and time.

2) AFFINITY CALCULATION

The affinity between antibodies and antigens indicates the ability of antibodies to solve antigens. Because the scheduling problem we want to solve is a multi-objective function optimization problem, the antigen corresponds to the objective

Algorithm 1 IMPSO Algorithm

Require: A DAG graph G transformed by a workflow
Ensure: A task-instance mapping solution *globalBestSol*

```
Initialize schedule components, the best gobal soltuion globalBestSol
for each particle i in population do
    initialize the speed and position
    Generate a solution sol_i for workflow
    Update globalBestSol
end for
while it doesn't meet iteration condition do
    Calculate the affinity of each particle
    Sort particles by affinity, and the best one is sent to the next new population without any operation
    for each particle in population except the best one do
        calculate its clone numbers
        for for each clones of particle i do
            crossing operation
            mutation operation
        end for
        update this particle
        reconstruction the solution and calculate the affnity
    end for
    choose the best N − 1 particles added to the new population.
    choose the best M particles to update memory unit
    generate d particles randomly to replace the worst d particles
    Update population and the bestGlobalSol
end while
```

function and constraint conditions to be optimized. Antibodies are candidate solutions. Therefore, the affinity between antibodies and antigens is expressed by the following formula:

$$\text{affinity}_i = (\alpha COST + \beta Makespan)/(dis_i + 1) \tag{12}$$

we consider the makespan and cost at the same time in the definition of affinity. In the optimization,the excellent degree of particles is judged by affinity. Therefore, the makespan and cost are measured at the same time. α and β are weight parameters, which are used for users to determine which parameter they are more concerned. If $\alpha = \beta = 0.5$, it means that cost is with the same importance to makespan. dis_i is the distance from the i-th particle to the current global best solution, which is calculated by the following formula.

$$\text{dis}_\text{i} = \sqrt{\sum_{i=1}^{N} (position_{ij} - gbest_j)^2} \tag{13}$$

$position_i$ and $gbest_i$ are the j-th dimension of $particle_i$ and global particle, respectively. It can be seen from the formula that the greater the fitness of the particle is and the closer it is to the optimal position, the greater the affinity of the particle is, and the smaller the affinity is.

In the process of affinity calculation, in order to speed up the convergence speed of the algorithm and ensure that the particles move in a better direction, we send the individuals with the highest affinity directly to the next generation without subsequent clone mutation operation.

The affinity between particles indicates the similarity between them, as shown in formula 14. The nom represents the Euclidean distance between antibody ab_i and ab_j. The smaller the distance between particles, the greater their affinity, indicating that they are more similar. Formula 14 is used to judge the similarity between two antibodies, so as to control the concentration of similar antibodies in the population. If the similarity in the population is too high, then it is easy to fall into local optimization in the subsequent optimization. Therefore, formula 14 is used to calculate the number of antibodies that are similar to antibody ab_i.

$$antibody_{i,j} = \exp(-nom(ab_i - ab_j)) \tag{14}$$

3) ANTIBODY CONCENTRATION AND INCENTIVE FUNCTION

When there are a large number of similar particles in the population, it is easy to fall into local optimization. It needs to be suppressed, and the concentration of antibody is defined by its affinity as shown in formula 15. Particles have a total of N dimensions.

$$\text{consistence}_i = \frac{\sum_{j=1}^{N} S_{i,j}}{N} \tag{15}$$

In which:

$$\text{S}_{i,j} = \begin{cases} 1, & consistence_{i,j}/\max\{consistence_{i,j}\} \geq \eta \\ 0, & consistence_{i,j}/\max\{consistence_{i,j}\} < \eta \end{cases} \tag{16}$$

η is antibody similarity coefficient, which is a constant. It can be seen from formula 12-16. Particles with high affinity with antigen and low concentration are more popular. Particle diversity and immature convergence can also be achieved. Therefore, the excitation function of the antibody is defined:

$$\text{Incentive}_\text{i} = affinity_i \times e^{-consistence_i} \tag{17}$$

4) ANTIBODY CLONING, CROSSING AND MUTATION OPERATIONS

According to the incentive, a separate proportional cloning operation is performed on each individual in the population, and the number of $particle_i$ cloned is

$$\text{num}_\text{i} = \frac{affinity_i}{\sum_{j=1}^{N} affinity_j} \times N \tag{18}$$

The greater the affinity of the individual, the more sub individuals will be cloned. This can protect good genes and speed up the convergence of the algorithm. In order to prevent falling into local optimization, we use the intermediate crossing and variation method based on real number to expand the population and increase the diversity of each individual except the male parent.

Intersection mode is middle crossing, and the child individual is produced according to the following formula 19.

$$child = parent_1 + \gamma \times (parent_2 - parent_1) \tag{19}$$

γ is a proportional factor generated by a uniformly distributed random number on [0, 1]. The value of each dimension of the child is calculated according to formula 19, each dimension has a new α value. Besides, if the memory unit is not empty, the parent particles are selected from it, otherwise, from the whole parent particle.

After the crossing operation of all particles, the roulette selection method is used to determine whether each particle needs to mutate, in which the particles with low intensive are more likely to be selected.

5) UPDATE POPULATION

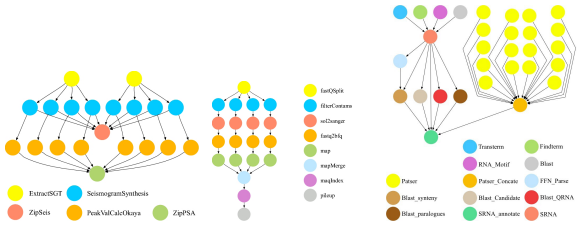
After all the operations, all particles are sorted according to the incentive, and the former N-1 particles are selected as the next generation population. It can be seen that the new population is composed of individuals with high incentive, which are obtained from their parent group. Some best particles are selected by a certain proportion to renew the memory unit. Furthermore, in order to screen out the worst particles, some new random generated particles will replace them. Then update the speed and position of all particles, and obtain a new global optimal solution. At this point, the iteration ends.

V. EVALUATION

In this section, we introduce the experimental environment and the setting we used and then compare the proposed methods with other approaches.

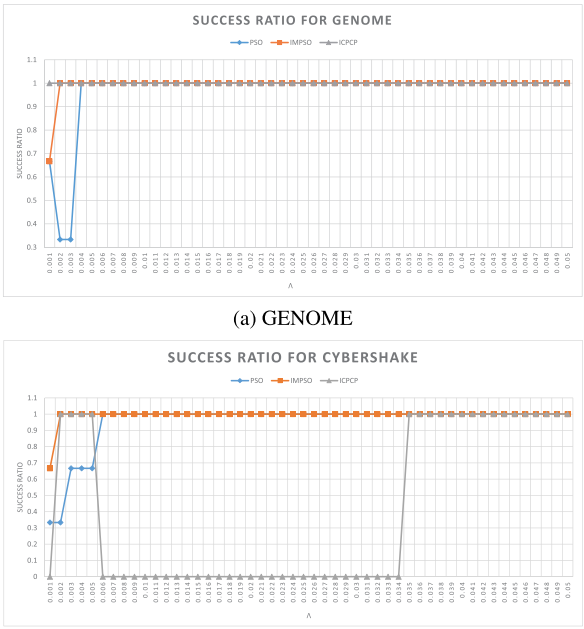
A. EXPERIMENT SETTINGS

We use some different scientific workflows for performance evaluation: GENOME, CYBERSHAKE and SIPHT. All of them have different focus on the requirements of the resource type. *Cybershake* [29] is a tool used by the Southern Californian Center (SCEC) to characterize the seismic hazard with the probabilistic seismic hazard analysis method. It is a data-intensive workflow that requires a large amount of memory and CPU. As shown in Fig. 4a. *Epigenomics* (genome) is a workflow application for bioinformatics to automate the sequencing of various genomes and is essentially a data processing pipeline. It is a CPU-intensive application, as depicted in Fig. 4b. *SIPHT* [30] is a workflow used by Harvard's bioinformatics project to automatically search databases for small untranslated RNA (sRNAs). That regulate bacterial secretion and toxicity It is a computationally intensive application, as shown in Fig. 4c. More details about these or more workflows can be found in [31]. Rafael Ferreira da Silva *et al.* [32] developed a workflow generator to produce synthetic workflow. Using this generator, we can get any size of scientific workflow in DAX (Directed Acyclic Graph in XML) format. Moreover, as a mature generator, its authors have created many sizes of different workflows for scientific



(a) Cybershake (b) Epigenomics (c) SIPHT

FIGURE 4. Structure of 3 kind of scientific workflows [31].



(a) GENOME

(b) CYBERSHAKE

FIGURE 5. Changed λ for influcing makespan of workflows.

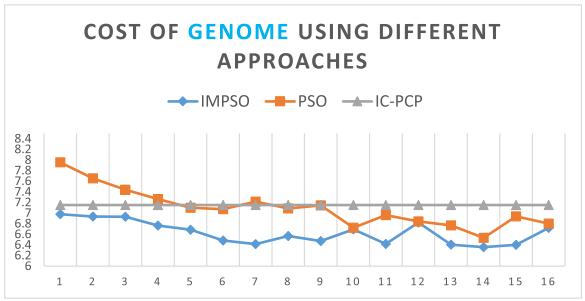
TABLE 1. Capabilities and costs of available instance types in [5].

TYPE	SPEED	COST	TYPE	SPEED	COST
$type_1$	1.0	0.12	$type_6$	3.5	0.595
$type_2$	1.5	0.195	$type_7$	4.0	0.72
$type_3$	2.0	0.28	$type_8$	4.5	0.855
$type_4$	2.5	0.375	$type_9$	5.0	1.0
$type_5$	3.0	0.48			

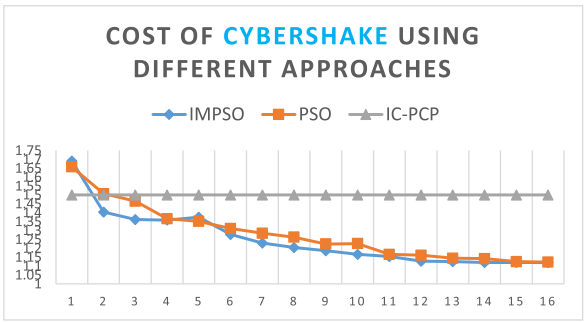
research to use. The workflows documents used as input in this paper are all provided by the [32].

Since it is difficult to perform repeatable experiments on real datacenters, we use simulation cloud environment imitated by [5]. we use actual Amazon cloud computing instance type parameters, as shown in Table 1. Refer to the parameters of Amazon EC2. The average bandwidth between instances is set to 20 MBps. The time interval to charge is set to one hour. In our experiment, we use the attribute ECU to represent the speed of instance. For simplicity and convenience, we take the speed of type 1 as the basic, and the speed of other types is relative speed. The unit of cost is dollar per hour.

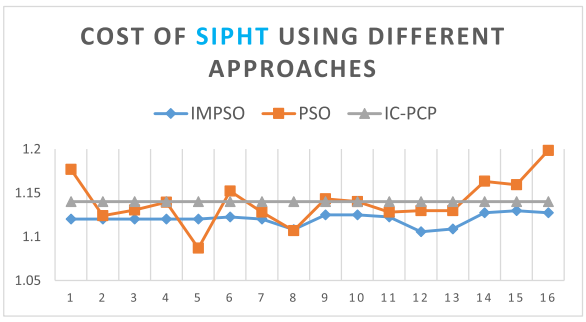
Our experiment is performed on a PC with Core (TM) i5 3.40FHz, 16GRAM, Windows 10, Java 2 Standard Edition V1.8.0.



(a) GENOME

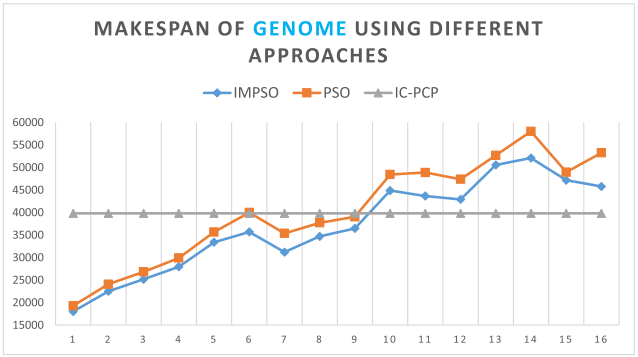


(b) CYBERSHAKE

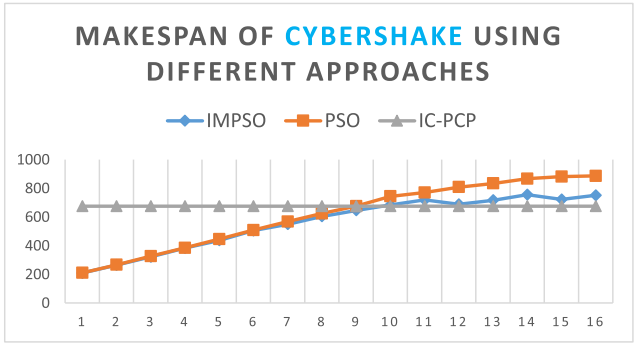


(c) SIPHT

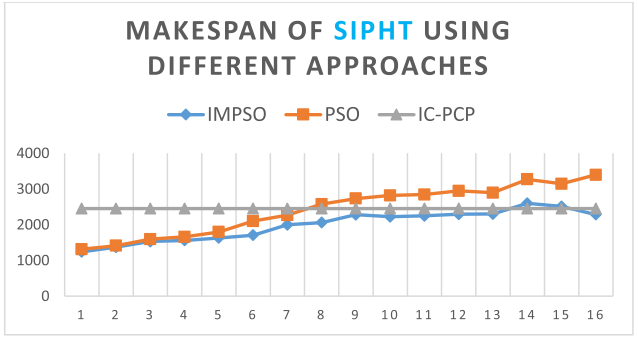
FIGURE 6. Cost for workflows with different approaches.



(a) GENOME



(b) CYBERSHAKE



(c) SIPHT

FIGURE 7. Makespan for workflows with different approaches.

B. EXPERIMENTAL RESULTS

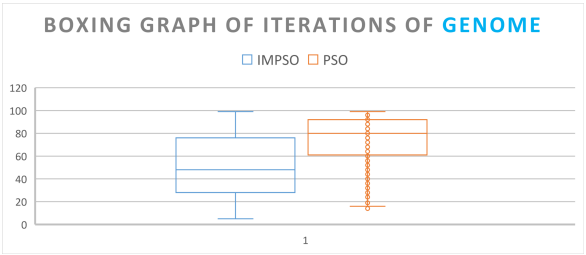
We compare our approach with PSO [26] and IC-PCP [25], which are one of the most cited works. These methods use time as a constraint to optimize the cost as much as possible. All methods have run 800 times. Because of the problem as two-objective optimization, and the characteristics of the proposed algorithm, the solutions obtained by each run may be different. Thus, In order to show the results more intuitively, we take averages of every 50 times in the 800 runs, and get 16 values drawed as figures.

According to analysis in [26], the learning factor and inertia factor of speed update in PSO are set to: $\omega = 0.5, c_1 = 2, c_2 = 2$. The size of population and iterations are set to 100. Combined with the research on immune mechanism and the results of experiments, the parameters related to immune mechanism are set as follows: the memory unit capacity is set to half of the population, and the number of randomly generated new particles are set to 1/10 of the population. The clone size was controlled at about twice the population.

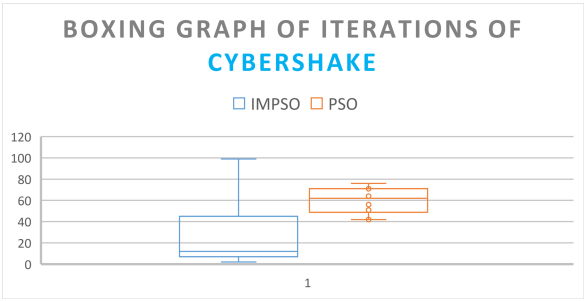
In addition, in order to evaluate the performance of the proposed algorithm in the heterogeneous environment such as cloud computing, we design two cases, both of which use a single example and use HEFT algorithm to determine the scheduling order to schedule. The first is that all tasks choose the cheapest instance type and run on the same instance. This scheme is the cheapest and the longest. The other is to select the fastest instance type for all tasks, which is often the most expensive. And each task has an instance. They're called *the slowest scheduling* and *the fastest scheduling*, respectively. These two scheduling methods are used to determine the upper and lower limits of the deadline. Besides, we chose a workflow of 50 for the experiment.

For evaluating our approaches that can meet the deadline constraint, we define a λ to control the loose degree of deadlines. We use the makespan D_{fast} of *the fastest scheduling* as the base deadline. And the range of deadlines is defined as:

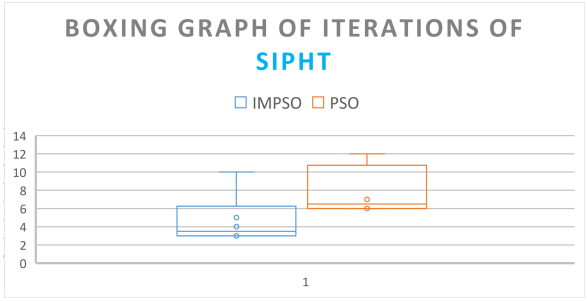
$$D = D_{fast} + \lambda \times (D_{slow} - D_{fast}) \tag{20}$$



(a) GENOME



(b) CYBERSHAKE



(c) SIPHT

FIGURE 8. Box graph of convergent algebra distribution.

in which $D_s low$ is the makespan of *the slowest scheduling*. We set the λ from 0.001 to 0.05, as 0.001 added. as depicted in Fig. 5. We can see that, for all workflow, IMPSO has better results than PSO. When λ is less than 0.002, IMPSO is possible to miss the solution in a round of optimization, while λ is less than 0.006 for PSO. For ICPCP, it is not stable. It performs worst in CYBERSHAKE, and when λ increases from 0.006 to 0.034, it can't find a solution that satisfies the deadline. For workflow SIPHT, the all three methods can obtain solution in each stage of deadline.

As regards to cost, the results show that the proposed IMPSO is better than the other two mothods for all the three workflows, which can be seen from Fig. 6. For all the 800 runs, we take an average for each method to compare. For workflow GENOME, the average costs of PSO and IC-PCP increased by 6.9% and 7.8%, respectively, compared with the proposed IMPSO. Similarly, the values are 1.8% for SIPHT, 2.4% and 2% for CYBERSHAKE.

For makespan, the results for different workflows are shown in Fig. 7. It can be seen that IMPSO is better than PSO in general, and most of the results are better than those of IC-PCP. In ICPCP, makespan is considered as constraint, and it optimize cost only. Thus, the makespan results of each run of IC-PCP algorithm are basically the same. The average value is taken to compare. For workflow SIPHT, the average makespans of PSO and IC-PCP increased by 21.8% and 47.8%, respectively, compared with IMPSO. Similarly, the values are 9.4% and 20.6% for CYBERSHAKE. For GENOME, IMPSO is slightly better the other two methods

Regarding convergence speed, IMPSO has great advantages over PSO. We run them 800 times respectively, and draw their convergence algebra in boxing graph as Fig. 8. As shown in the figure, IMPSO has faster convergence speed no matter which workflow is scheduled.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose an immune-based particle swarm optimization (IMPSO) algorithm used to schedule workflow in cloud environment. The objective is to minimize the execution cost and makespan under user-defined deadline constraint. Experimental evaluation shows that the proposed IMPSO is competitive with other solutions, which can obtain a better makespan with a lower cost increasingly. The results also show that this algorithm can obtain the best solution at a faster convergence period.

In future work, we are interested in clustering the workflow tasks before allocating them to cloud resources, which may introduce further challenges because of the dependency between tasks and trade-off between clusters.

REFERENCES

[1] M. Wang, G. Liu, P. Zhao, C. Yan, and C. Jiang, "Behavior consistency computation for workflow nets with unknown correspondence," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 281–291, Jan. 2018.

[2] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on Petri nets with data operations," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 251–260, Jan. 2018.

[3] X. Guo, S. Wang, D. You, Z. Li, and X. Jiang, "A siphon-based deadlock prevention strategy for S3PR," *IEEE Access*, vol. 7, pp. 86863–86873, 2019.

[4] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S 4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.

[5] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, "Deadline-constrained cost optimization approaches for workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3401–3412, Dec. 2017.

[6] P. Wang, C. Zhao, and Z. Zhang, "An ant colony algorithm-based approach for cost-effective data hosting with high availability in multi-cloud environments," in *Proc. IEEE 15th Int. Conf. Netw., Sens. Control (ICNSC)*, Zhuhai, China, Mar. 2018, pp. 1–6.

[7] P. Wang, W. Zhou, C. Zhao, Y. Lei, and Z. Zhang, "A dynamic programming-based approach for cloud instance types selection and optimization," *Int. J. Inf. Technol. Manage.*, to be published.

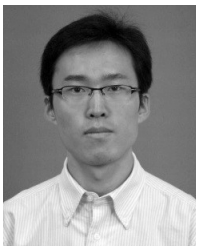
[8] W. Liu, P. Wang, Y. Meng, Q. Zhao, C. Zhao, and Z. Zhang, "A novel model for optimizing selection of cloud instance types," *IEEE Access*, vol. 7, pp. 120508–120521, 2019.

[9] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments: Workflow scheduling algorithms for clouds," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 8, p. e4041, Apr. 2017.

[10] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Future Gener. Comput. Syst.*, vol. 52, pp. 1–12, Nov. 2015.

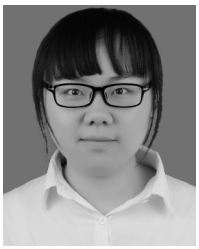
[11] S. Kaur, P. Bagga, R. Hans, and H. Kaur, "Quality of service (QoS) aware workflow scheduling (WFS) in cloud computing: A systematic review," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 2867–2897, Apr. 2019.

[12] J. A. J. Sujana, T. Revathi, T. S. S. Priya, and K. Muneeswaran, "Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing," *Soft Comput.*, vol. 23, no. 5, pp. 1745–1765, Mar. 2019.
[13] M. Wieczorek, A. Hoheisel, and R. Prodan, "Towards a general model of the multi-criteria workflow scheduling on the grid," *Future Gener. Comput. Syst.*, vol. 25, no. 3, pp. 237–256, Mar. 2009.
[14] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," *SIGMOD Rec.*, vol. 34, no. 3, p. 56, Sep. 2005.
[15] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proc. 8th Heterogeneous Comput. Workshop (HCW)*. Washington, DC, USA: IEEE Computer Society, Jan. 1999, p. 30.
[16] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task scheduling strategies for workflow-based applications in grids," in *Proc. 5th IEEE Int. Symp. Cluster Comput. Grid (CCGrid)*, vol. 2. Washington, DC, USA: IEEE Computer Society, 2005, pp. 759–767.
[17] A. Radulescu, A. J. C. V. Gemund, and H.-X. Lin, "Llb: A fast and effective scheduling algorithm for distributed-memory systems," in *Proc. 13th Int. Symp. Parallel Process. (IPPS), 10th Symp. Parallel Distrib. Process. (SPDP)*. Washington, DC, USA: IEEE Computer Society, Jan. 1999, pp. 525–530.
[18] J. Ullman, "NP-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.
[19] Y.-K. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms," in *Proc. 5th Merged Int. Parallel Process. Symp. Symp. Parallel Distrib. Process.*, Nov. 2002, p. 531.
[20] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, "Scheduling precedence graphs in systems with interprocessor communication times," *SIAM J. Comput.*, vol. 18, no. 2, pp. 244–257, Apr. 1989.
[21] 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.
[22] R. Sundarrajan, V. Vasudevan, and S. Mithya, "Workflow scheduling in cloud computing environment using firefly algorithm," in *Proc. Int. Conf. Electr., Electron., Optim. Techn. (ICEEOT)*, Mar. 2016, pp. 955–960.
[23] A. Mohan, M. Ebrahimi, S. Lu, and A. Kotov, "Scheduling big data workflows in the cloud under budget constraints," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 2775–2784.
[24] H. Jiang, H. E, and M. Song, "Dynamic scheduling of workflow for makespan and robustness improvement in the IaaS cloud," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 4, pp. 813–821, 2017.
[25] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.
[26] 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. 2014.
[27] M. S. Kumar, I. Gupta, and P. K. Jana, "Delay-based workflow scheduling for cost optimization in heterogeneous cloud system," in *Proc. 10th Int. Conf. Contemp. Comput. (IC3)*. Los Alamitos, CA, USA: IEEE Computer Society, Aug. 2017, pp. 1–6, doi: 10.1109/ic3.2017.8284323.
[28] B. Fang and L. Sun, "Cloud workflow scheduling optimization oriented to QoS and cost-awareness," (in Chinese), *Comput. Integr. Manuf. Syst.*, vol. 24, no. 2, pp. 331–348, Feb. 2018.
[29] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, D. Okaya, P. Small, and K. Vahi, "CyberShake: A physics-based seismic hazard model for southern California," *Pure Appl. Geophys.*, vol. 168, nos. 3–4, pp. 367–381, Mar. 2011.
[30] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs," *PLoS ONE*, vol. 3, no. 9, p. e3197, Sep. 2008.
[31] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, Mar. 2013.
[32] R. F. D. Silva, W. Chen, G. Juve, K. Vahi, and E. Deelman, "Community resources for enabling research in distributed scientific workflows," in *Proc. IEEE 10th Int. Conf. e-Science*, Oct. 2014, pp. 177–184.



PENGWEI WANG received the B.S. and M.S. degrees in computer science from the Shandong University of Science and Technology, Qingdao, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2013.

He finished his Postdoctoral Research work at the Department of Computer Science, University of Pisa, Italy, in 2015. He is currently an Associate Professor with the School of Computer Science and Technology, Donghua University, Shanghai. His research interests include cloud computing, service computing, and data mining.



YINGHUI LEI received the B.S. degree in information security from Donghua University, Shanghai, China, in 2017, where she is currently pursuing the master's degree in computer science and technology. Her current research interests include cloud computing, workflow, and query optimization.

PROMISE RICARDO AGBEDANU received the B.Sc. degree in ICT from the Presbyterian University College, Ghana, in 2014, and the M.Phil. degree in information technology from the Kwame Nkrumah University of Science and Technology, Ghana, in 2018. He is currently pursuing the master's degree in computer science and technology with Donghua University, Shanghai, China. His current research interests include computer security, network security, and cloud security. He is also interested in using machine and deep learning to solve cyber-security problems.

ZHAOHUI ZHANG received the B.S. degree in computer science from Anhui Normal University, Wuhu, China, in 1994, the master's degree from the University of Science and Technology of China, in 2000, and the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2007. He was a Professor with Anhui Normal University, before July 2015. He became a Teacher at Anhui Normal University. He is currently a Professor with the School of Computer Science and Technology, Donghua University, Shanghai. His research interests include network information services, service computing, and cloud computing.