IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 12, 2019, accepted September 28, 2019, date of publication October 8, 2019,
date of current version October 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946216

An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing

SHANCHEN PANG[1], WENHAO LI[1], HUA HE[2], ZHIGUANG SHAN[3], AND XUN WANG[1]
[1]College of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China
[2]School of Mathematics and Statistics, Shandong University of Technology, Zibo 255049, China
[3]Informatization and Industry Development Department, State Information Center, Beijing 100045, China

Corresponding author: Hua He (huahe@sdut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572522, Grant 61572523, and Grant 61672033, and in part by the Key Research and Development Program of Shandong Province under Grant 2017GGX10147.

ABSTRACT As one of the hot issues in cloud computing, task scheduling is an important way to meet user needs and achieve multiple goals. With the increasing number of cloud users and growing demand for cloud computing, how to reduce the task completion time and improve the system load balancing ability have attracted increasing interest from academia and industry in recent years. To meet the two aforementioned goals, this paper develops an EDA-GA hybrid scheduling algorithm based on EDA (estimation of distribution algorithm) and GA (genetic algorithm). First, the probability model and sampling method of EDA are used to generate a certain scale of feasible solutions. Second, the crossover and mutation operations of GA are used to expand the search range of solutions. Finally, the optimal scheduling strategy for assigning tasks to virtual machines is realized. This algorithm has advantages of fast convergence speed and strong search ability. The algorithm proposed in this paper is compared with EDA and GA via the CloudSim simulation experiment platform. The experimental results show that the EDA-GA hybrid algorithm can effectively reduce the task completion time and improve the load balancing ability.

INDEX TERMS Cloud computing, task scheduling, task completion time, load balancing.

I. INTRODUCTION

In recent years, cloud computing has become a hot research topic, and it is widely used in telecommunications, manufacturing, education and scientific research [1], [2]. For example, storage clouds [3] provide secure data storage, backup and recording services, which provide great convenience for users. Educational clouds [4] can virtualize various types of hardware education resources and then transmit them to the internet system, providing a convenient information platform for education departments, teachers and students. In cloud computing, resources such as hardware, software and platforms are provided as services with the ''pay-as-you-go'' model. Users need to pay for only the services or resources they need without having to purchase hardware infrastructure. The current studies focus on virtualization, resource management, cloud security, green computing, task scheduling, and so forth. As cloud computing services rapidly grow, how to effectively schedule tasks to computational resources (virtual machines) according to goals has become increasingly important.

The goals of task scheduling mainly include reducing task completion time and energy consumption and improving resource utilization and load balancing ability [5]–[7]. With the dramatic increase in the number of cloud users, reducing task completion time is helpful for improving user experience. Improving load balancing ability contributes to fully utilizing virtual machines to prevent execution efficiency from decreasing due to the overload of resources or waste caused by excessive idle resources [8]–[10]. However, the above two objectives are mutually constrained. For instance, to reduce task completion time, it is easy to centrally schedule the tasks on the resources with strong computing power, which will cause a load imbalance problem. Therefore, it is challenging to design and optimize the task scheduling algorithm to balance the two goals of reducing completion time and improving load balancing ability.

The task scheduling problem has been proven to be NP-complete, and the optimal solution cannot be obtained in limited time [11], [12]. For a problem, if the guess of a solution can be verified in polynomial time and the time

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

to solve the problem is considered to rapidly increase as the size of the problem increases, the current computing approaches cannot be used to determine an accurate answer in meaningful time. In this case, the problem can be considered NP-complete. Topcuouglu *et al.* [13] proposed that the task scheduling problem is NP-complete in the general case, as well as in some restricted cases, such as scheduling tasks with one or two times to two processors and scheduling unit-time tasks to an arbitrary number of processors. Ilavarasan and Thambidurai [14] proposed this problem in the most general case, which has been proven to be NP-complete for which optimal solutions can be found only after an exhaustive search. At the same time, intelligent model design of complex systems is a key issue for organization responsiveness to uncertainties. The model of task scheduling in cloud computing is a complex intelligent model that contains a large number of tasks and heterogeneous computing resources [15]–[17].

To date, evolutionary algorithms have solved many scheduling and mapping problems. EDA [18], [19] is a population-based evolutionary algorithm that has been proven to be effective in solving many optimization problems. The probability model can describe the distribution of the solutions in the search space, and new solutions are generated by sampling it. EDA has fast convergence speed and can find a good solution in a short time; however, EDA can easily fall into a local optimum. GA [20], [21] is an algorithm that simulates the natural selection and genetic mechanism of biological evolution. This algorithm is a parallel and global search method that provides a general framework for solving complex system optimization problems. The selection, crossover and mutation operations can expand the search range of solutions. GA is characterized by great global search ability that effectively compensates for the deficiency of EDA, but its convergence speed is slow. Inspired by the successful applications of EDA and GA and by comparing their advantages and disadvantages, an EDA-GA hybrid algorithm is proposed to provide an effective strategy for multi-objective task scheduling in cloud computing.

The main contributions of this paper are summarized as follows.

- First, this paper proposes a multi-objective task scheduling model that defines the demands of the tasks for virtual machines in detail. This model regards scheduling performance and time as the constraints of the scheduling problem and achieves the multiple objectives for reducing task completion time and improving load balancing ability.
- Second, we propose an EDA-GA hybrid algorithm to solve the multi-objective task scheduling problem. This paper innovatively applies EDA to task scheduling problems, and a combination of EDA and GA has been used to help us find the optimal solution.
- Finally, this paper verifies the effectiveness of the proposed EDA-GA hybrid algorithm through comparative experiments. Using the CloudSim simulation experiment platform, we compare and analyze EDA-GA, EDA and GA for the goal of this paper. The experimental results show that the EDA-GA hybrid algorithm is an efficient multi-objective task scheduling algorithm in cloud computing.

The remainder of this paper is organized as follows. The related work on this problem is introduced in Section II. The system model and mathematical model for the task scheduling problem are provided in Section III. The EDA-GA hybrid algorithm to solve this problem is proposed in detail in Section IV. Experiments and analysis are shown in Section V. Finally, the paper is ended with some conclusions and future work in Section VI.

II. RELATED WORK

In the cloud computing environment, the key to task scheduling is to find the optimal mapping relation between tasks and virtual machines according to the goals of users and cloud systems. The main methods to solve this problem include single-objective optimization algorithms and multi-objective optimization algorithms [22]–[24].

A. SINGLE-OBJECTIVE OPTIMIZATION ALGORITHMS

Single-objective optimization algorithms mainly apply traditional scheduling algorithms such as Min-Min [25], Max-Min [26], and Sufferage [27]. On this basis, some improvements have been made. Wu *et al.* [28] proposed a segmented Min-Min algorithm in which the tasks were first ordered by their expected completion time. Then, the ordered sequence was segmented, and Min-Min was applied to these segments. This algorithm worked better than Min-Min when the lengths of the tasks were dramatically different by giving longer tasks the chance to be executed earlier than when the original Min-Min was adopted. Etminani and Naghibzadeh [29] proposed a new scheduling algorithm based on Min-Min and Max-Min and selected between the two algorithms based on the standard deviation of the expected completion time of tasks on virtual machines. The experimental results showed that the new algorithm could lead to significant performance gains for a variety of scenarios. Devipriya and Ramesh [30] proposed an improved Max-Min algorithm based on the expected execution time of tasks, scheduling large tasks to virtual machines with lower computing speed and scheduling small tasks to virtual machines with high computing speed, which could effectively reduce the completion time of the overall task. Traditional scheduling algorithms generally have low adaptability and extensibility. For example, the Min-Min algorithm starts with small tasks and assigns tasks to the most efficient resources in turn. The strategy is easy to perform, and although it effectively reduces the task completion time, it leads to load imbalance. The long-term overload operation of high-computational-efficiency virtual machines will not meet the quality of service needs [31], [32].

B. MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

To improve the shortcomings of single-objective optimization algorithms, multi-objective optimization algorithms

have been proposed. Multi-objective optimization algorithms mainly use swarm intelligence algorithms such as EDA, GA, and ACO (ant colony optimization) [33], [34], which can find a near-optimal solution for a complex scheduling problem within a certain time. Gupta and Garg [35] proposed the LB-ACO algorithm, which used the ACO approach to obtain local optimal solutions. Finally, nondomination sorting was applied to obtain the Pareto set of solutions representing the trade-off between the makespan time and load balancing in the cloud. Liu *et al.* [36] proposed a task scheduling algorithm based on a genetic ant colony algorithm that used the strong global search capability of GA to obtain a better solution and then converted the solution into the initial pheromone of ACO to finally obtain optimal scheduling through the positive feedback of the ACO. Cui and Zhang [37] proposed a workflow task scheduling algorithm based on GA. This algorithm designed a two-dimensional coding method and a new genetic crossover and mutation operation to produce new offspring to increase the population diversity. Li et al. [38] proposed the use of modified ACO in load balancing. This method balanced the entire system load while attempting to minimize the makespan of a given task set. Xiao *et al.* [39] proposed a task scheduling scheme based on the sharing mechanism and swarm intelligence optimization algorithms. Combining ACO, GA and the ABC (artificial bee colony) algorithm, a sharing module was designed to share the optimal solutions found by the three algorithms and then continued to explore the solution space. This combination of methods accelerated the convergence of the algorithm and improved the convergence accuracy. Wu and Wang [40] proposed an improved EDA algorithm to solve the parallel scheduling problem of tasks with priority constraints. A probability model was designed to determine the relative position of tasks to satisfy the priority constraints among tasks, and the scheduling scheme with the shortest completion time and the lowest energy consumption was gradually found. Aziza and Krichen [41] used GA to model and optimize the task scheduling problem. The results showed that the algorithm performed well in terms of cost and completion time. Li et al. [42] proposed a multi-objective task scheduling GA-DE algorithm based on GA and the DE (differential evolution) algorithm, which introduced DE into the mutation stage of GA to give full play to the global search ability of GA, taking advantage of its fast convergence speed to reduce the time it takes for the algorithm to produce an optimal solution. The results showed that the algorithm was superior to GA and DE in terms of quality of service and load balancing. Singh *et al.* [43] provided a valuable survey of the algorithms for cloud computing. This work will enable us to determine a suitable approach for recommending better schemes for scheduling users' applications. Chiaraviglio *et al.* [44] proposed an approach for jointly optimizing the two objectives of power consumption minimization and maintenance cost minimization in cloud optimization. Chen et al. [45] proposed a novel multi-objective ant colony system based on co-evolutionary multiple populations, which adopted two colonies to address the two objectives of optimizing both the execution time and execution cost.

Most of the existing methods do not consider load balancing factors when attempting to reduce the task completion time. In addition, EDA has not been well used to solve the task scheduling problem. In this paper, an EDA-GA hybrid algorithm is proposed to solve the multi-objective task scheduling problem with the criteria of reducing task completion time and improving load balancing ability.

III. MODEL

A. SYSTEM MODEL

Users submit tasks to the cloud system, and the cloud system includes three modules: task manager, resource manager and scheduler [46]. The cloud system sends tasks to the task manager, which processes tasks in batch mode and obtains information such as the sizes of tasks. The resource manager uniformly manages all virtual machines and obtains information such as the computing speeds of virtual machines. After obtaining information such as the sizes of tasks submitted by the task manager and the computing speeds of virtual machines submitted by the resource manager, the scheduler starts working. The scheduler is the core component and is responsible for allocating tasks to virtual machines using the EDA-GA hybrid algorithm proposed in this paper. The structure of the task scheduling mechanism in cloud computing is shown in Fig. 1.

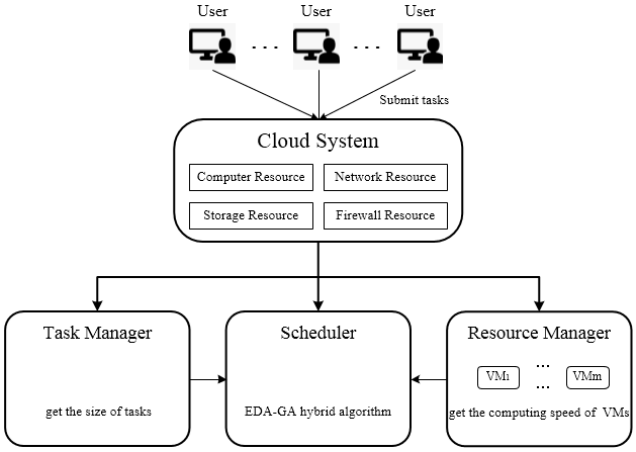


FIGURE 1. Structure of task scheduling mechanism in cloud computing.

B. MATHEMATICAL MODEL

The multi-objective task scheduling problem in this paper can be described as follows. There are n tasks to be scheduled on m virtual machines with different computing speeds. Each task can be scheduled on any virtual machine. Each virtual machine can execute multiple tasks. The multi-objective optimization problem considers more than one objective simultaneously to find a trade-off between the conflicting objectives. Our aim is to map tasks to all virtual machines to reduce the task completion time and improve the load balancing ability.

1) MATHEMATICAL MODEL NOTATION

The main notations of this paper are listed in Table 1.

TABLE 1. Main notation definition.

Notation	Definition
n	The number of tasks
m	The number of virtual machines
T_i	The task i
V_j	The virtual machine j
TS_i	The size of task i
VS_j	The computing speed of virtual machine j
$T = \{T_1, T_2, \ldots, T_n\}$	The set of tasks
$V = \{V_1, V_2, \ldots, V_m\}$	The set of virtual machines
$TS = \{TS_1, \ldots, TS_n\}$	The task size set
$VS = \{VS_1, \ldots, VS_m\}$	The virtual machine computing speed set
$ETC_{n\times m}$	$ETC_{n\times m}$ matrix of size $n \times m$, represents the running time of all tasks on each virtual machine
$x_{i,j,r}$	$\{0, 1\}$. $x_{i,j,r} = 1$ illustrates that task i is the r-th task processed on virtual machine j; otherwise, $x_{i,j,r} = 0$

2) SYSTEM FEASIBILITY CONSTRAINTS

This paper assumes that tasks submitted by users are independent of each other. There are no constraints or communication between tasks. Some system feasibility constraints are defined as follows [47].

$$\sum_{j=1}^{m}\sum_{r=1}^{n} x_{i,j,r} = 1, \quad i = 1, 2, \ldots, n \tag{1}$$

$$\sum_{i=1}^{n} x_{i,j,r} \le 1, \quad j = 1, 2, \ldots, m; \ \forall r \tag{2}$$

$$\sum_{i_1=1}^{n} x_{i_1,j,r+1} - \sum_{i_2=1}^{n} x_{i_2,j,r} \le 0, \quad j = 1, 2, \ldots, m; \ \forall r \tag{3}$$

where formula (1) guarantees that a task can only be scheduled on one virtual machine and only once; formula (2) ensures that each virtual machine processes no more than one task at a time; and formula (3) means that tasks on a certain virtual machine are scheduled in order.

3) TASK COMPLETION TIME MODEL

The task size set and the virtual machine computing speed set are known; then, the *ETC* matrix can be calculated according to formula (4).

$$ETC(t_i, r_j) = \frac{TS_i}{VS_j} (1 \le i \le n, 1 \le j \le m) \tag{4}$$

where $ETC(t_i, r_j)$ represents the time required for task i to complete running on virtual machine j.

The virtual machine's completion time is the sum of the running time to execute all the tasks assigned to it. The completion time of each virtual machine can be calculated as formula (5).

$$time_j = \sum_{r=1}^{k} ETC(t_r, r_j) \tag{5}$$

where k is the total number of tasks assigned to virtual machine j.

Due to the parallel computing in cloud computing, this paper defines the total completion time as the maximum completion time of all virtual machines. The total task completion time is calculated as formula (6).

$$CompleteTime = max\{time_1, time_2, \ldots, time_m\} \tag{6}$$

4) LOAD BALANCING MODEL

Another goal of this paper is to improve system load balancing ability. The load balancing degree is defined as formula (7).

$$DBL = \frac{\sum_{j=1}^{m} time_j}{m \times CompleteTime} \tag{7}$$

where *DBL* represents the load balancing degree. For the virtual machine's completion time, the completion time of each virtual machine is almost the same as the total completion time, which means that the load is more balanced. Therefore, the larger the *DBL* is, the more balanced the load and the stronger the load balancing ability.

5) FITNESS FUNCTION

Each individual in the population produced by the EDA-GA hybrid algorithm represents a feasible solution to the problem. The fitness function is used to evaluate the quality of solutions. It is the key to avoiding falling into a local optimum and achieving the optimal solution. We can build different fitness functions according to different requirements. This paper takes the total task completion time and load balancing degree into account. The fitness function is defined as formula (8) and formula (9).

$$GValue = \omega_1 * \frac{1}{CompleteTime} + \omega_2 * DBL \tag{8}$$

$$\omega_1 + \omega_2 = 1 (0 \le \omega_1, \omega_2 \le 1) \tag{9}$$

where ω_1 and ω_2 are weight coefficients. Different weight coefficients can be set according to different user requirements. For example, if only considering the task completion time factor, set ω_1 to 1 and ω_2 to 0. If only the load balancing factor is considered, set ω_1 to 0 and ω_2 to 1. This paper considers the above two factors simultaneously and sets ω_1 and ω_2 to 0.5 and 0.5, respectively. The larger the *GValue* is, the better the quality of the solution.

IV. EDA-GA HYBRID ALGORITHM

The EDA-GA hybrid algorithm is designed as follows: first, use EDA to initialize the probability model. During initialization, all the probabilities are set to $1/m$, and the roulette method is used for sampling to generate a certain scale of solutions. At the same time, according to *GValue*, evaluate all

solutions and choose a number of excellent solutions. Second, use GA to perform crossover and mutation operations on the selected excellent solutions and generate new solutions. Third, evaluate the excellent solutions from step 1 and the new solutions in step 2 and sort them in descending order. The top $p\%$ of the excellent solutions are selected to form the elite population. Finally, update the probability model according to the elite population. Run the algorithm in such a way until the stopping condition is met and output the optimal solution. The specific process of the EDA-GA hybrid algorithm is as follows.

A. OPERATIONS OF EDA

1) INITIALIZATION

The probability model describes the distribution of the solutions. Better solutions are easier to obtain if the characteristics of the problem can be reflected by the model. The probability model is designed as follows.

$$P(g) = \begin{bmatrix} p_{11}(g) & p_{12}(g) & \cdots & p_{1m}(g) \\ p_{21}(g) & p_{22}(g) & \cdots & p_{2m}(g) \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1}(g) & p_{n2}(g) & \cdots & p_{nm}(g) \end{bmatrix} \tag{10}$$

$P(g)$ represents the mapping relationship between n tasks and m virtual machines in the g-th iteration. During initialization, all probability values are set to $1/m$ to ensure the randomness of the initial population.

2) SAMPLING METHOD

We generate the population by sampling the probability model. It uses a roulette method to generate the population of size PS. Each individual in the population represents a solution to assign tasks to virtual machines.

Individuals are coded by the indirect coding method. Suppose that there are 5 virtual machines and 10 tasks; the individual generated after sampling is as follows.

$$\{1, 4, 2, 3, 2, 5, 1, 2, 3, 4\}$$

This coding method encodes the virtual machines occupied by each task. The length of each individual is equal to the number of tasks. Each position in the individual represents the task number. The value in this position represents the virtual machine number assigned to the task. This individual represents the first task assigned to the first virtual machine, the second task assigned to the fourth virtual machine, and the third task assigned to the second virtual machine.

3) FITNESS ASSESSMENT

For all individuals generated in the last step, according to the coding result of each individual, the distribution of tasks on virtual machines can be obtained. Taking the individual in the last step as an example, the individual can be decoded as shown in Table 2.

According to Table 2, the completion time of each virtual machine can be obtained by formula (5). Then, the fitness value of the individual is obtained. All the individuals are evaluated according to the fitness value and sorted in descending order. Finally, the top 50 % of excellent individuals are selected.

TABLE 2. Task-VM allocation table.

VM	Task
1	1,7
2	3,5,8
3	4,9
4	2,10
5	6

B. OPERATIONS OF GA

The crossover operation and mutation operation are the main genetic methods of GA. They can effectively expand the search range of solutions and increase the diversity of the population to achieve the optimal solution. After obtaining the excellent individuals in the last step, the GA algorithm is run to perform crossover and mutation operations on them to generate a new population. Then, the excellent individuals in the last step and the newly generated individuals in this step are evaluated according to $GValue$. The individuals are sorted in descending order, and the top $p\%$ individuals are selected to form the elite population. At the same time, the local optimal solution is obtained.

In crossover, some positions of two parent individuals are exchanged to produce two child individuals, as shown in Fig. 2. We use 1-point crossover, which is performed in a random manner.

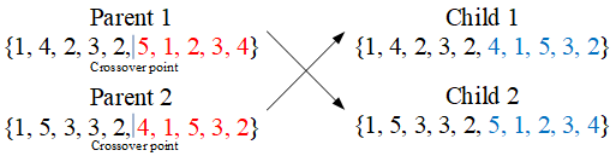


FIGURE 2. Crossover.

In mutation, we use the method of exchange mutation to randomly select two positions and exchange the values in the two positions, which is shown in Fig. 3.

FIGURE 3. Mutation.

This step simultaneously evaluates the fitness of the excellent individuals in the last step and the newly generated individuals in this step to prevent the loss of the excellent individuals.

C. UPDATING METHOD

The probability model is updated using the elite population and the PBIL (population-based incremental learning) method [48] as follows.

$$p_{ij}(g+1) = (1-\lambda)p_{ij}(g) + \lambda\frac{1}{E}\sum_{k=1}^{E} I_{ij}^{k}(g) \tag{11}$$

$$I_{ij}^{k}(g) = \begin{cases} 1, & if\ T_i\ on\ V_j\ in\ the\ k-th\ individual \\ 0, & else \end{cases} \tag{12}$$

where $P_{ij}(g)$ is the probability that task i is assigned to virtual machine j in the g-th iteration, $\lambda \in (0, 1)$ denotes the learning rate, E denotes the size of elite population ($E = PS \times p\%$), and $I_{ij}^{k}(g)$ is an indicator function that corresponds to the k-th individual of the elite population.

The implementation process of the EDA-GA hybrid algorithm is shown as pseudocode in Algorithm 1.

Algorithm 1 EDA-GA Hybrid Algorithm

Input: Set of tasks T, set of virtual machines V
Output: A solution for scheduling tasks to virtual machines

```
Initialize the probability model
while iter <= iter_max do
  Sample the probability model to generate the population
  for each individual in the population do
    Calculate GValue according to formula (5)
    Sort all individuals in descending order according to
    GValue
    Select the top 50% of the excellent individuals,
    denoted as TEP
    for each individual in TEP do
      Perform crossover operation
      Perform mutation operation
      Express the new population produced in the above
      two steps as TNP
    end for
    for each individual in TEP and TNP do
      Calculate GValue according to formula (5)
      Sort all individuals in descending order according
      to GValue
      Select the top p% of the excellent individuals to
      form the elite population
    end for
  end for
  Use the elite population to update the probability model
  iter++
end while
```

V. EXPERIMENTS AND ANALYSIS

To test the effectiveness of the proposed EDA-GA hybrid algorithm, we compare it with EDA and GA based on CloudSim. CloudSim is a cloud computing simulation software announced by Grid Laboratory at the University of Melbourne and Gridbus project in April 2009. The primary objective is to quantify and compare the scheduling strategy for different service and application models on cloud infrastructure [49]. The experiment is run on a PC with a 2.50 GHz processor and 4 GB RAM.

CloudSim is used to randomly generate the task size set and the virtual machine computing speed set within the limits of real-world cloud environments to simulate a wide variety of tasks submitted by users and to allow a cloud host to be shared concurrently among multiple virtual machines with varying performance. The EDA-GA scheduling strategy is placed on DatacenterBroker, which is responsible for mediating between users and the service provider according to users' requirements. In a real cloud computing environment, cloud application developers can generate a combination of user request distribution, application configuration, and cloud availability scenarios through the user code layer in the stack, and perform reliable tests based on custom cloud configurations already supported in CloudSim [49].

A. PARAMETER SETTINGS

To fairly compare the proposed EDA-GA with EDA and GA, the stopping criterion of these three algorithms is set to 100 generations because the results are basically unchanged after the 100-th iteration. According to the characteristics of GA, the crossover rate and mutation rate are set to 0.8 and 0.05, respectively. EDA-GA contains three important parameters: PS (size of the population), E (size of the elite population, $E = PS \times p\%$), and λ (learning rate). To determine the values of the above three parameters, the Taguchi design-of-experiment (DOE) method [50] is used to analyze the influence of the parameters on the performance of the algorithm under different parameter levels.

TABLE 3. Factor levels of parameters.

Parameters	Factor levels			
	1	2	3	4
PS	100	150	200	250
p	10	20	30	40
λ	0.05	0.10	0.15	0.20

In this experiment, four levels are set for each parameter, as shown in Table 3. The orthogonal array $L_{16}(4^3)$ is chosen accordingly, and a moderate instance is used (the number of tasks is set to 1000 and the number of virtual machines is set to 10). For each parameter combination, the completion time is used as a standard, the EDA-GA is run 10 times and the average value is used as the response value (RV), as shown in Table 4.

The influence trend of each parameter is shown in Figs. 4 - 6.

From Figs. 4 - 6, $PS = 100$, $p = 30$, and $\lambda = 0.1$ are recommended parameter settings. The balance of search depth and breadth is affected by PS, and it is appropriate to choose 100. For the percentage of elite population, the results

TABLE 4. The RV value.

Number	Factor			RV
	PS	p	λ	
1	1	1	1	2.1831
2	1	2	2	2.1827
3	1	3	3	2.1975
4	1	4	4	2.1992
5	2	1	2	2.2119
6	2	2	1	2.2057
7	2	3	4	2.2202
8	2	4	3	2.2203
9	3	1	3	2.2198
10	3	2	4	2.2220
11	3	3	1	2.2205
12	3	4	2	2.2115
13	4	1	4	2.2321
14	4	2	3	2.2332
15	4	3	2	2.1981
16	4	4	1	2.2222

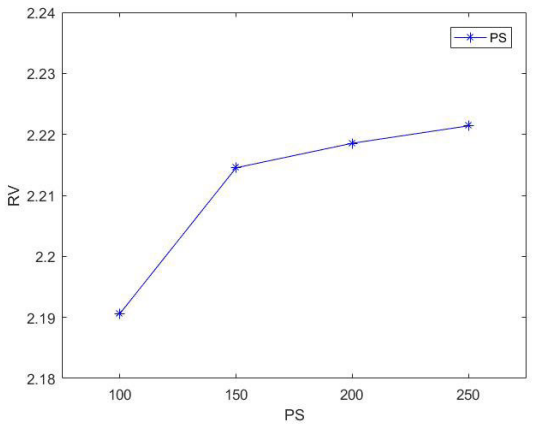


FIGURE 4. The influence trend of *PS*.

indicate that it has less influence. If the value of p is small, it may lead to failure to update the probability model effectively because the useful information from the elite population is insufficient, and if the value of p is large, it may reduce the accuracy of the model because worse solutions are brought into the elite population. Here, we choose 30% as the percentage of the elite population. The learning rate affects the convergence speed of the algorithm, and $\lambda = 0.1$ is recommended.

B. EXPERIMENTAL RESULTS

1) EXPERIMENT 1

To verify the superiority of the proposed EDA-GA hybrid algorithm, we compare it with EDA and GA using the same

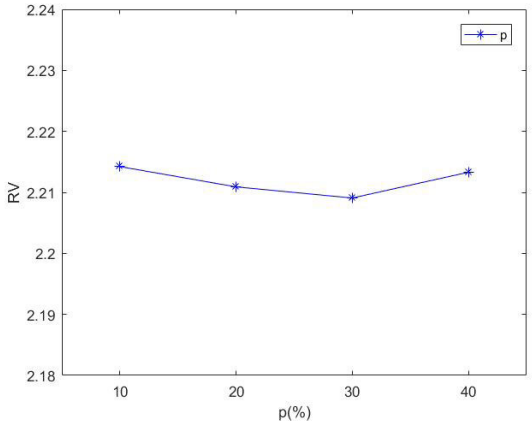


FIGURE 5. The influence trend of *p*.

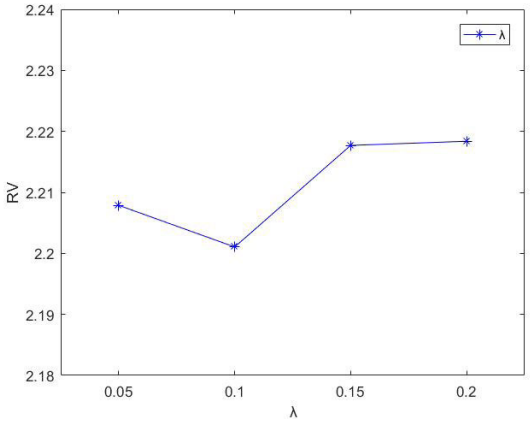


FIGURE 6. The influence trend of λ.

parameter settings as described in the previous section. The experimental testing is performed in three different instances:

Instance 1: A few small tasks along with many large tasks;

Instance 2: A few large tasks along with many small tasks;

Instance 3: The sizes of tasks are randomly determined.

We choose 1000 as the number of tasks and 10 as the number of virtual machines. These values are fixed throughout the three instances. The results and analysis focus on three aspects: task completion time, load balancing degree and fitness value.

A. Test of the task completion time: *CompleteTime*

As shown in Figs. 7 - 9, EDA-GA performs better in each instance in terms of task completion time. In the three instances, EDA-GA has an average reduction of 3.2% and 15.1% compared to EDA and GA, respectively.

B. Test of the load balancing: *DBL*

The results in Figs. 10 - 12 show that EDA-GA has higher *DBL* values than the other two algorithms. Compared with EDA and GA, EDA-GA has an average increase of 7.6% and 11.7%, which means that EDA-GA has advantages in terms of load balancing.

C. Test of the fitness value: *GValue*

As shown in Figs. 13 - 15, the fitness value of EDA-GA is the highest among the three algorithms, and it is 5.4% and

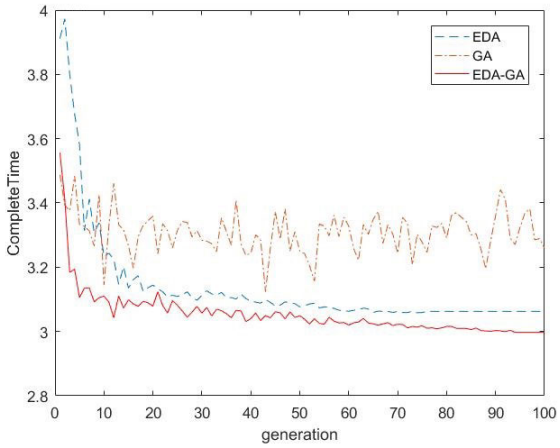


FIGURE 7. *CompleteTime* of instance 1.

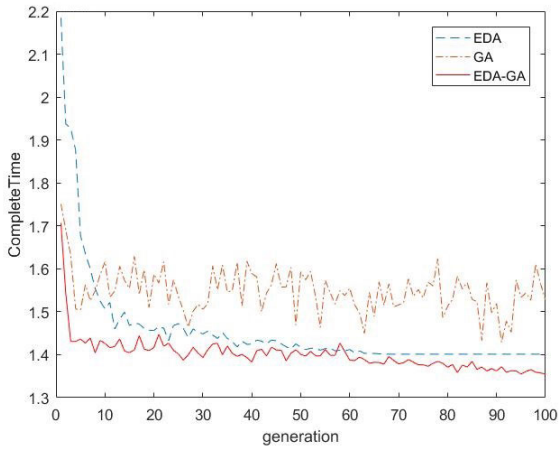


FIGURE 8. *CompleteTime* of instance 2.

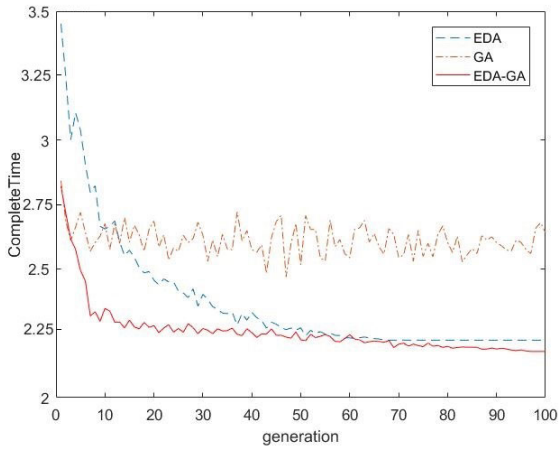


FIGURE 9. *CompleteTime* of instance 3.

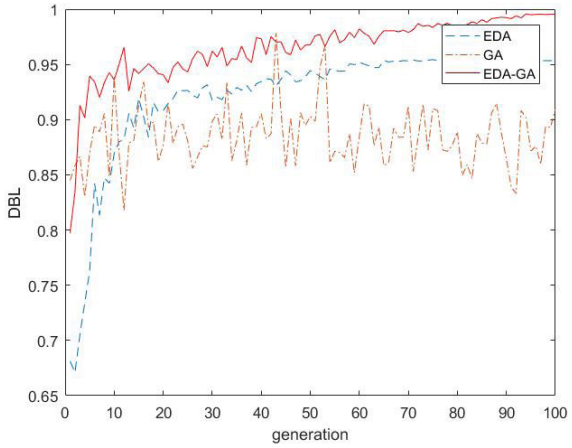


FIGURE 10. *DBL* of instance 1.

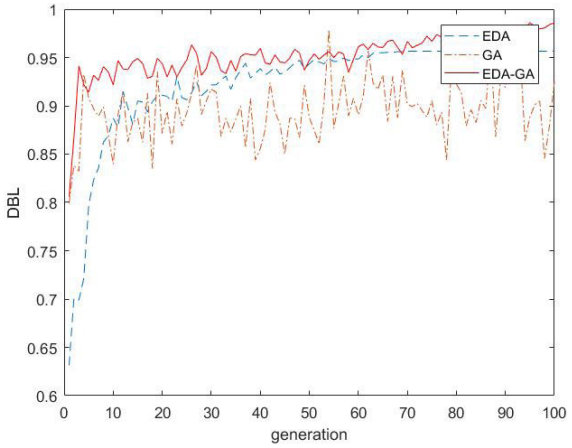


FIGURE 11. *DBL* of instance 2.

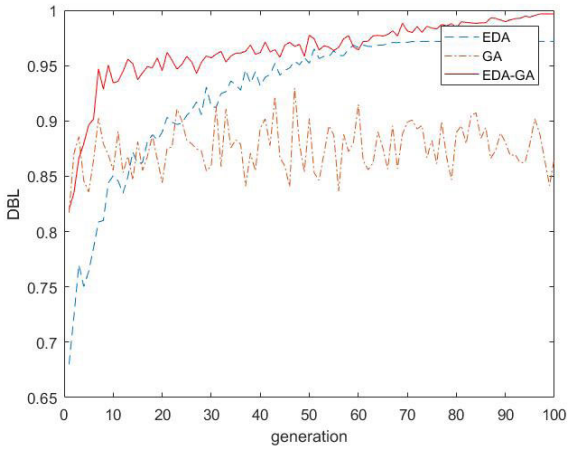


FIGURE 12. *DBL* of instance 3.

9.1% higher than the other two algorithms. This result is more in line with the goal of this paper, which is to find a trade-off between reducing task completion time and improving load balancing ability.

In conclusion, the experimental results show that the EDA-GA hybrid algorithm has a lower task completion time, more balanced load and higher fitness value than the other two algorithms under different instances. The reason mainly lies in the fact that the sampling mechanism and probability

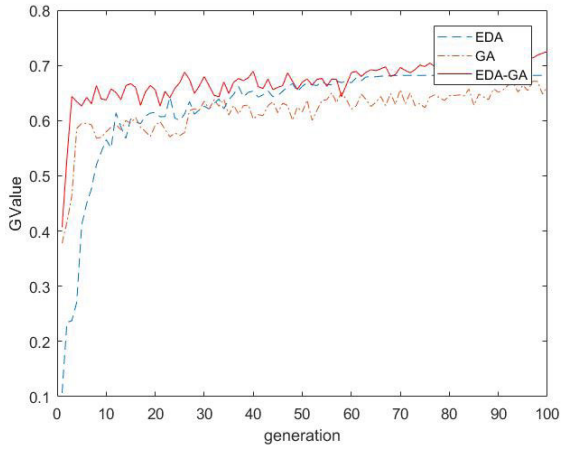


FIGURE 13. ***GValue*** **of instance 1.**

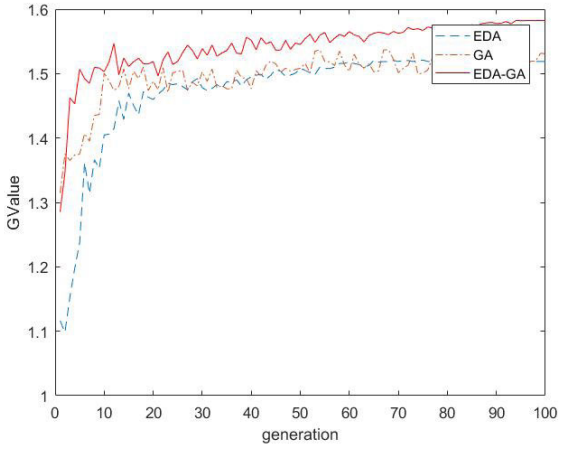


FIGURE 14. ***GValue*** **of instance 2.**

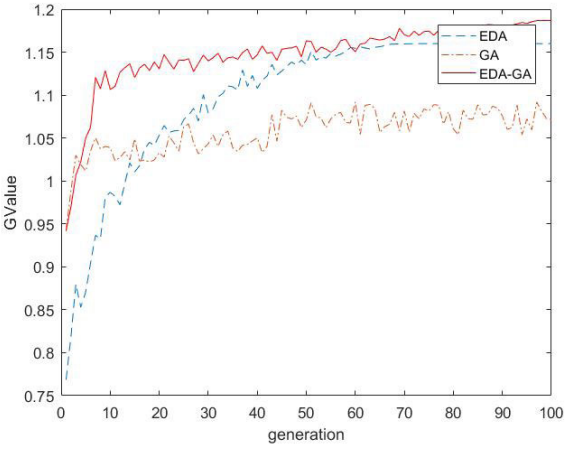


FIGURE 15. ***GValue*** **of instance 3.**

model contribute to finding feasible and excellent solutions quickly. Moreover, the crossover and mutation operations expand the range of the solutions to prevent the algorithm from falling into a local optimum. Based on the above analysis, the EDA-GA hybrid algorithm proposed in this paper has better performance.

2) EXPERIMENT 2

To verify the effectiveness of the proposed algorithm under different numbers of tasks, an experiment is conducted with reference to the task completion time. In this test, the number of virtual machines is set to 10, and we analyze the changes in task completion time when the number of tasks is 100, 200, . . . , 1000. The results are shown in Fig. 16.

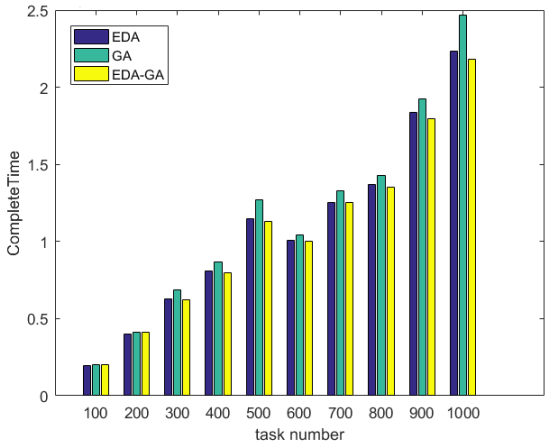


FIGURE 16. **Three algorithms' CompleteTime for different numbers of tasks.**

The results in Fig. 16 show that when the number of tasks is small, the task completion time of the three algorithms is almost the same. However, as the number of tasks increases, the algorithm proposed in this paper has a lower completion time. This is more suited to handle tasks in a real cloud computing environment in which the number of tasks is large.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose an effective EDA-GA hybrid algorithm to address the multi-objective task scheduling problem with the goal of reducing the task completion time and improving the system load balancing ability. The algorithm first uses the operations of EDA to generate some feasible solutions, then uses the operations of GA to generate new solutions based on the excellent solutions selected in the previous step to expand the search range of solutions, and finally, it finds the optimal solution. We evaluate the proposed algorithm by comparing it with EDA and GA on CloudSim. The results show that the proposed EDA-GA hybrid algorithm has good convergence speed and search ability, and it performs better in reducing task completion time and improving load balancing ability. However, this paper does not consider the dynamics and uncertainty of the cloud computing environment. On the one hand, the computing speed of virtual machines changes in real time. On the other hand, virtual machines can join or exit the cloud system at any time.

Future work will focus on task scheduling issues that are closer to those in real cloud computing environments.

There may be priority constraint relationships between tasks. In addition, in terms of the objective, cost is an important factor that affects task scheduling in real life. If users want to reduce the task completion time, they need to spend more money purchasing cloud computing resources. We would like to design a task scheduling algorithm that balances the three factors of task completion time, cost and load balancing.

REFERENCES

[1] S. K. Panda and P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *J. Supercomput.*, vol. 71, no. 4, pp. 1505–1533, Jan. 2015.
[2] N. Xiong, A. V. Vasilakos, J. Wu, Y. R. Yang, A. Rindos, Y. Zhou, W.-Z. Song, and Y. Pan, "A self-tuning failure detection scheme for cloud computing service," in *Proc. 26th Parallel Distrib. Proces. Symp. (IPDPS)*, Washington, DC, USA, May 2012, pp. 668–679.
[3] N. Vurukonda and B. T. Rao, "A study on data storage security issues in cloud computing," *Proc. Comput. Sci.*, vol. 92, pp. 128–135, Dec. 2016.
[4] B. Wang and H. Y. Xing, "The application of cloud computing in education informatization," in *Proc. Int. Conf. Comput. Sci. Service Syst. (CSSS)*, Nanjing, China, Jun. 2011, pp. 2673–2676.
[5] F. Zhang, J. Cao, K. Li, S. U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Gener. Comput. Syst.*, vol. 37, pp. 309–320, Jul. 2014.
[6] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," in *Mobile Networks and Applications*. New York, NY, USA: Springer, 2019, pp. 1–11.
[7] H. Gao, H. Miao, L. Liu, J. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: A visualization transform tool perspective," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 10, pp. 1369–1397, 2018.
[8] L. Bai, Y.-L. Hu, S.-Y. Lao, and W.-M. Zhang, "Task scheduling with load balancing using multiple ant colonies optimization in grid computing," in *Proc. IEEE 6th Int. Conf. Natur. Comput. (ICNC)*, Yantai, China, Aug. 2010, pp. 2715–2719.
[9] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, "Collaborative service selection via ensemble learning in mixed mobile network environments," *Entropy*, vol. 19, no. 7, p. 358, 2017.
[10] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.
[11] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.
[12] H. Gao, S. Mao, W. Huang, and X. Yang, "Applying probabilistic model checking to financial production risk evaluation and control: A case study of Alibaba's Yu'e Bao," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 785–795, Sep. 2018.
[13] 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.
[14] E. Ilavarasan and P. Thambidurai, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *J. Comput. Sci.*, vol. 3, no. 2, pp. 94–103, Feb. 2007.
[15] J. Li, N. Xiong, J. H. Park, C. Liu, S. Ma, and S. Cho, "Intelligent model design of cluster supply chain with horizontal cooperation," *Future Gen. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.
[16] H. Cheng, Z. Su, N. Xiong, and Y. Xiao, "Energy-efficient node scheduling algorithms for wireless sensor networks using Markov random field model," *Inf. Sci.*, vol. 329, pp. 461–477, Oct. 2015.
[17] Y. Yin, W. Xu, Y. Xu, H. Li, and L. Yu, "Collaborative QoS prediction for mobile service with data filtering and slopeone model," *Mobile Inf. Syst.*, vol. 2017, pp. 1–14, Jun. 2017.
[18] P. Larraanaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, MA, USA: Kluwer, 2001.
[19] J. Sun, Q. Zhang, and E. P. K. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," *Inf. Sci.*, vol. 169, nos. 3–4, pp. 249–262, 2005.
[20] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *J. Parallel Distrib. Comput.*, vol. 70, no. 1, pp. 13–22, Jan. 2010.
[21] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Sci. Program.*, vol. 14, nos. 3–4, pp. 217–230, 2006.
[22] C. Wu, L. Wang, and X. Zheng, "An effective estimation of distribution algorithm for solving uniform parallel machine scheduling problem with precedence constraints," in *Proc. Int. Conf. Evolut. Comput. (CEC)*, Vancouver, BC, Canada, Jul. 2016, pp. 2626–2632.
[23] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.
[24] H. Gao, Y. Duan, H. Miao, and Y. Yin, "An approach to data consistency checking for the dynamic replacement of service process," *IEEE Access*, vol. 5, pp. 11700–11711, 2017.
[25] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Proc. Comput. Sci.*, vol. 57, pp. 545–553, Dec. 2015.
[26] Y. Mao, X. Chen, and X. Li, "Max-min task scheduling algorithm for load balance in cloud computing," in *Proc. Int. Conf. Comput. Sci. Inf. Technol. (CSAIT)*, Kunming, China, 2014, pp. 457–465.
[27] X. Kong, M. Lin, Y. Jiang, W. Yan, and X. Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1068–1077, Jul. 2011.
[28] M.-Y. Wu, W. Shu, and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems," in *Proc. 9th Heterogeneous Comput. Workshop (HCW)*, Washington, DC, USA, Jan. 2000, pp. 375–385.
[29] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorihtm for grid task scheduling," in *Proc. 3rd IEEE/IFIP Int. Conf. Central Asia Int. (ICI)*, Tashkent, Uzbekistan, Oct. 2007, pp. 1–7.
[30] S. Devipriya and C. Ramesh, "Improved max-min heuristic model for task scheduling in cloud," in *Proc. Int. Conf. Green Comput. (ICG)*, Chennai, India, Dec. 2013, pp. 883–888.
[31] E. K. Tabak, B. B. Cambazoglu, and C. Aykanat, "Improving the performance of independentTask assignment heuristics MinMin, MaxMin and sufferage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1244–1256, May 2014.
[32] Y. Yin, S. Aihua, G. Min, X. Yueshen, and W. Shuoping, "QoS prediction for Web service recommendation with network location-aware neighbor selection," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, no. 4, pp. 611–632, 2016.
[33] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *Proc. 8th Int. Conf. Comput. Eng. Syst. (ICCES)*, Cairo, Egypt, Nov. 2013, pp. 64–69.
[34] H. Gao, D. Chu, and Y. Duan, "The probabilistic model checking based service selection method for business process modeling," *J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 6, pp. 897–923, Aug. 2017.
[35] A. Gupta and R. Garg, "Load balancing based task scheduling with ACO in cloud computing," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Doha, UAE, Sep. 2017, pp. 174–179.
[36] C. Y. Liu, C. M. Zou, and P. A. Wu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing," in *Proc. 13th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci. (DCABES)*, Xian Ning, China, Nov. 2014, pp. 68–72.
[37] Y. Cui and X. Zhang, "Workflow tasks scheduling optimization based on genetic algorithm in clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Chengdu, China, Apr. 2018, pp. 6–10.
[38] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. 6th Annu. Chinagrid Conf.*, Liaoning, China, Sep. 2011, pp. 3–9.
[39] F. U. Xiao, "Task scheduling scheme based on sharing mechanism and swarm intelligence optimization algorithm in cloud computing," *Comput. Sci.*, vol. 45, no. 1, pp. 303–307, 2018.
[40] C.-G. Wu and L. Wang, "A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system," *J. Parallel Distrib. Comput.*, vol. 117, pp. 63–72, Jul. 2018.
[41] H. Aziza and S. Krichen, "Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing," *Computing*, vol. 100, no. 2, pp. 65–91, Feb. 2018.
[42] Y. Li, S. Wang, X. Hong, and Y. Li, "Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm," in *Proc. 37th Chin. Control Conf. (CCC)*, Wuhan, China, Jul. 2018, pp. 4489–4494.

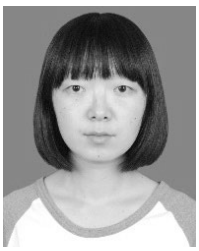
[43] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 1–51, Apr. 2017.
[44] L. Chiaraviglio, F. D'Andreagiovanni, R. Lancellotti, M. Shojafar, N. Blefari-Melazzi, and C. Canali, "An approach to balance maintenance costs and electricity consumption in cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 4, pp. 274–288, May 2018.
[45] Z.-G. Chen, Z.-H. Zhan, Y. Lin, Y.-J. Gong, T.-L. Gu, F. Zhao, H.-Q. Yuan, X. Chen, Q. Li, and J. Zhang, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.
[46] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.
[47] M.-A. H. Abdel-Jabbar, I. Kacem, and S. Martin, "Unrelated parallel machines with precedence constraints: Application to cloud computing," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Luxembourg, Luxembourg, Oct. 2014, pp. 438–442.
[48] H. Li, S. Kwong, and Y. Hong, "The convergence analysis and specification of the population-based incremental learning algorithm," *Neurocomputing*, vol. 74, no. 11, pp. 1868–1873, May 2011.
[49] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.
[50] D. C. Montgomery, *Design and Analysis of Experiments*. New York, NY, USA: Wiley, 1984.

SHANCHEN PANG received the graduation degree from the Tongji University of Computer Software and Theory, Shanghai, China, in 2008. He is currently a Professor with the China University of Petroleum, Qingdao, China. His current research interests include theory and application of Petri nets, service computing, and trusted computing.

WENHAO LI received the B.S. degree from the Shandong University of Science and Technology, Qingdao, China, in 2017. She is currently pursuing the degree with the China University of Petroleum, Qingdao. Her current research interests include cloud computing and task scheduling.

HUA HE received the Ph.D. degree in computer application technology from Tianjin University. She is currently a Lecturer with the Shandong University of Technology. Her current research interests include Petri nets, big data processing, and performance analysis methods.

ZHIGUANG SHAN received the B.A. degree in automation engineering and the Ph.D. degree in computer science from the University of Science and Technology Beijing, Beijing, China, in 1997 and 2002, respectively. He is currently a Professor and the Director of the Informatization and Industry Development Department, State Information Center of China. His current research interests include computer networks, performance evaluation, strategic planning and top design of smart city, macro planning, and developing policies of informatization.



XUN WANG received the graduation degree from the Guangxi Normal University of Mathematical Science, Guilin, China, in 2010. She is currently an Associate Professor with the China University of Petroleum, Qingdao, China. Her current research interests include wisdom medical and bioinformatics.