Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 13, no. 8, December 2017, pp. 1206-1218
DOI: 10.23940/ijpe.17.08.p4.12061218

An Approach to Resource Scheduling based on User Expectation in Cloud Testing

Zhongsheng Qian*, Xiaojin Wang

School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China

Abstract

Cloud testing, with the features of automatic deployment, parallel submission, on-demand distribution and timely response, has been widely favored by many users. Therefore, it is crucial to reduce energy consumption, satisfy user requirement and timely response to user requests for resources, which are guaranteed by a good resource scheduling scheme. The requirements and benefits between user and provider of cloud testing are comprehensively measured in this work. On one hand, in order to meet the expectations of different users for the finish time and cost of their tasks, the definition of user expectation is introduced and then a dynamic pricing model is constructed to achieve the flexible conversion between time and cost. On the other hand, genetic algorithm is employed to implement resource scheduling in cloud testing, which can shorten the running time of all tasks on the cloud testing platform to improve the efficiency and reduce the load as greatly as possible. Finally, comparative experiments show that the scheme proposed in this work is feasible and efficient.

Keywords: cloud testing; resource scheduling; user expectation; genetic algorithm; pricing model

(Submitted on October 8, 2017; Revised on November 6, 2017; Accepted on November 23, 2017)

1. Introduction

With the increment of software scale and complexity, software testing is playing a more and more important role in software development. The cost of manual testing is too high and there are also many problems with automated testing. For example, the resources needed for software testing are not readily available, and it takes huge amounts of materials to build and dismantle the testing environment for each testing [15]. Besides, the hardware and software resources for testing, whose utilization rate is not high, are left unused after testing. In addition, the results of software testing require manual statistics, which is time-consuming.

Cloud computing, which has the characteristics of automatic resource generation and flexible allocation to greatly reduce the cost of automated testing, provides new possibilities for automated testing. Cloud computing with the characteristics of flexible services, resource pooling, on-demand services, affordable services, and ubiquitous access serves to build more reliable automated test environments and provide more convenient and low-cost software testing services [14]. Users can easily obtain automated testing services on demand and no longer need to prepare testing environment. In cloud testing platform [11], users can submit the testing tasks and testing scripts via the browser. Then, the testing tasks are allocated to cloud resources by scheduling methods. Finally, the testing results are sent to the users. All in all, cloud testing serves to reduce the resource cost for users and provides efficient support for complex testing environment [12].

The rapid development of cloud computing has brought new challenges and opportunities to IT industry, and it also makes cloud testing a new model of business services [5]. However, with the development of technology and changing demands, the number and complexity of testing task is increasing. In addition, cloud testing is favoured by more and more users owing to its automatic deployment, parallel submission, on-demand allocation and timely service. Therefore, the

* Corresponding author.
E-mail address: changesme@163.com

pressure of cloud testing platform surges, which leads to the importance of how to improve the efficiency of cloud testing. Moreover, a suitable resource scheduling strategy can not only improve test efficiency, but also reduce energy consumption.

In recent years, there has been a lot of research about resource scheduling. For example, several different methods are used to reduce energy consumption and total running time of all tasks in cloud platform [4,18,19,27]. However, there are few studies that concurrently take into account the completion time and cost to meet user demands from users' perspective. Moreover, cloud testing platform, a business service model that users need to pay to use, often employs the pay-as-you-go mechanism [8]. In fact, there is research about how to charge for users' tasks. The authors [10,22,25,26], for example, presented different pricing methods for users' tasks from different perspectives, but all of them only take the execution time of tasks into account. However, if the cost of tasks cannot change with other factors, it is obvious that users hope that their tasks are finished as early as possible, which can cause a task running queue to jam during rush hour.

The load pressure of the cloud platform mainly concentrates on the working hours while there are many free resources in off-work time. In fact, there are only some real-time tasks that have a high requirement for time and also some tasks with a low time requirement. Therefore, we can use a new model that realizes the conversion of time and cost, which can appropriately extend the completion time of tasks that have a low requirement for time to relieve the load pressure of cloud platform during the peak period.

In view of the above problems, we propose the concept of user expectation, and construct a new dynamic pricing model and resource scheduling scheme. The concrete work is illustrated as follows:

- A new dynamic pricing model, which is different from traditional pricing models that only depend on the execution time of tasks, is designed. And the parameter *waiting time* is introduced. In addition, the cost will decrease accordingly when the waiting time increases in the dynamic model.
- The concept of user expectation is proposed to meet different users' demands as greatly as possible by combining the two aspects of the user's demand for time and cost.
- The resource scheduling scheme based on user expectation is explored. Besides, considering the requirements of cloud testing provider, genetic algorithm is employed to minimize the completion time of all tasks and reduce the energy consumption.

The remainder of this paper is structured as follows: in Section 2, the methods and results of previous studies on resource scheduling in cloud platform are summarized. In Section 3, a detailed introduction is given to explain the framework of the proposed method. In Section 4, the experimental setup and results are analyzed. The conclusion and future work are given in section 5.

2. Related work

With the development of cloud computing, resource scheduling has been a hot topic. Considering the temporal and spatial patterns of resource requirements and resource failures, G. Tian, et al. [19] proposed node resource provision policies based on failure rules for heterogeneous services consolidated in cloud computing infrastructure to ensure the reliability of dynamically provided resources. D. Ergu, et al. [4] presented a model, by which the computing resources can be allocated according to the rank of tasks, for task-oriented resource allocation in a cloud computing environment. X. Shi, et al. [18] designed a cloud utility maximization model to schedule the cloud resource, and subgradient algorithm is used to solve Lagrangian relaxation dual problem to maximize the resource utility. In order to meet SLA (Service Level Agreement) and minimize resource cost, X. Zhao, et al. [27] explored a resource allocation model based on service selection in cloud that transforms component services, responding performance and resource costs into candidate logical service set in resource scheduling. However, [4,18,19,27] contribute to maximizing the efficiency and revenue of provider from the perspective of cloud platform providers without taking into account the actual needs of users.

W. Wei, et al. [23] proposed a fast resource placement based on nonlinear programming theory. In order to meet regionally distributed demands for various resources in large-scale online services, the service providers need to decide where to place resources to satisfy massive demands from all regions. Aiming at the irrational resource allocation problem in the Xen virtualization platform, B. Qiao, et al. [17] presented two resource scheduling optimization algorithms to improve resource utilization efficiency, avoid unnecessary virtual machine migrations and balance the load between virtual machines. Z. Xiao, et al. [24] combined the skewness metric with virtual machines with different resource characteristics to dynamically allocate the resources and optimize the number of servers in use to reduce the cost. C. Papagianni, et al. [16] designed a unified resource allocation framework, in which the optimal networked cloud mapping problem is formulated as a mixed integer programming (MIP for short) problem to indicate objectives related to cost efficiency of the resource

mapping procedure while abiding by user requests for QoS-aware virtual resources. Considering the allocation of cloud resources and energy consumption, the above studies aim at meeting the needs of the overall platform users and service quality, but hardly guarantee the needs of each user.

Cloud platform is a new business service model, in which a pricing scheme is required for tasks submitted by users. Therefore, how to achieve the resource pricing scheme has been a hot topic [3]. P. Xiao, et al. [25] explored a hybrid gaming-based pricing model is proposed to overcome the demerits of existing price mechanisms in terms of efficiency and fairness. A. K. Kar, et al. [10] proposed a flexible pricing approach in order to maximize the revenue and address users' diverse requirements systematically. X. Wu, et al. [22] designed a dynamic pricing model, which consists of the *k*-means algorithm and Bayes decision algorithm, to maximize the revenue of cloud service providers. Considering the competing characteristics under multi-tenant environment in cloud computing and aiming to improve the profit of both the resource supply and demand sides, Y. Yuan, et al. [26] presented an uncompleted information game-based cloud resource allocation model and a dynamical pricing game model based on the predicted bid value to achieve maximum benefits. S. Chaisiri, et al. [3] proposed an optimal cloud resource provisioning algorithm by formulating a stochastic programming model to address the uncertainty of consumer's future demand and providers' resource prices. In general, the above studies serve to construct a pricing model for resource allocation, which only considers users' request mode and response time of cloud providers. However, when a dynamic pricing model is constructed, other factors including the characteristics of tasks submitted by users and users' requirements for completion time and cost of tasks, also need to be taken into account.

Resource scheduling in cloud testing has been an NP-hard problem. To address this problem, many heuristic algorithms are applied to optimize the process of resource scheduling. Y. Zheng, et al. [28] introduced VM scheduling strategies based on artificial intelligence to save energy, balance load and improve QoS performance. J. T. Tsai, et al. [20] explored an improved differential evolution algorithm based on the proposed cost and time models in cloud computing environment to optimize task scheduling and resource allocation. J. Huang [7] proposed a GA-based workflow task scheduling scheme to improve the user experience and maximize the economic benefits. Y. Zheng, et al. [29] focused on the dependencies between testing tasks and present Ant Colony Optimization based on testing task dependencies to achieve the resource scheduling in cloud testing. In fact, various heuristic algorithms have been used to optimize cloud resource scheduling. In addition, the characteristics of cloud computing are considered to improve the performance of cloud computing.

3. Resource scheduling in cloud testing

Cloud testing can provide remote testing services by deploying multiple virtual machines on the server. Therefore, users do not need to purchase and install automated testing tools and deploy testing environment on local physical hosts, which can reduce users' testing cost and provide much convenience for users. If users want to finish testing tasks in cloud testing platform, they only need to submit testing tasks and pay the fee, and then wait for the testing report to download. The whole testing process is simple and easy. However, when there are a number of tasks to execute in cloud testing platform, it is very important that virtual machines are reasonably allocated to handle tasks. Besides, VM scheduling has been an NP-hard problem, it is necessary to design an appropriate VM scheduling scheme.

Figure 1 briefly describes the entire flow of VM resource scheduling. Each user can submit multiple tasks on cloud test platform while these tasks are allocated to virtual machine to execute through scheduling algorithm.

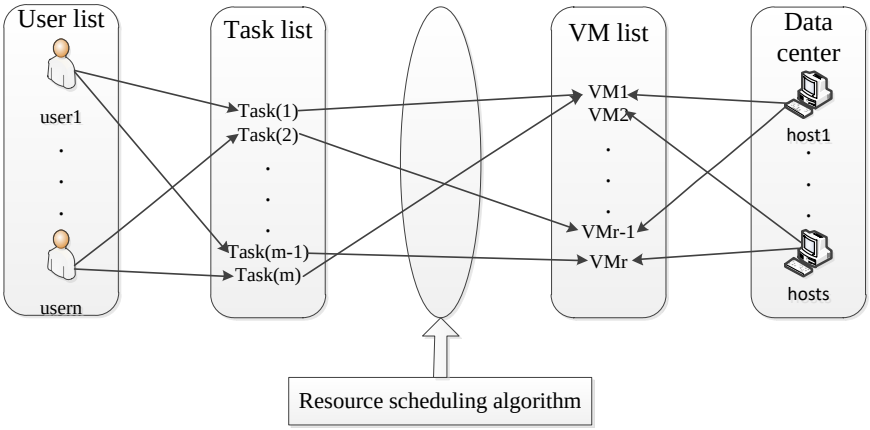

Figure 1. The framework of resource scheduling

In this work, we propose an approach to resource scheduling based on user expectation in cloud testing and design a dynamic pricing model to meet each user's requirement as greatly as possible and make the task finished in a shorter time and lower fee.

3.1. User expectation

User expectation is represented by the time and cost required by users who submit a task on the cloud testing platform. In order to ensure quality of service, user expectation should be met when VM resources are allocated to tasks.

During a certain time period, there are many tasks submitted by users to be executed while the number of virtual machine resources is limited in cloud testing platform, which makes it difficult to finish all tasks in time. In fact, some tasks must be finished in time owing to a high real-time requirement while others do not have an urgent time requirement. Therefore, an approach to trade cost for time is proposed to assign the real-time tasks with higher priority, as well as delay the tasks with a lower time requirement and reduce their cost accordingly.

Suppose there is a task queue $\{task_1, task_2, \ldots, task_{n-1}, task_n\}$, and users' time and cost requirements for n tasks are $\{ET_1, ET_2, \ldots, ET_{n-1}, ET_n\}$ and $\{EC_1, EC_2, \ldots, EC_{n-1}, EC_n\}$ respectively. Then, we use a triple $\{task_i, ET_i, EC_i | i \in \{1, 2, \ldots, n\}\}$ to represent users' expectation for each task.

Cloud platform is a new business service model, which uses a pay-as-you-go scheme to charge for users' tasks. However, the traditional static pricing mechanism only considers the characteristic of tasks, which cannot meet users' requirements well. Therefore, a new dynamic pricing model is designed to trade cost for time, which makes users' cost change with the waiting time in task queue.

3.2. Dynamic pricing model

Suppose there are a task queue $\{task_1, task_2, \ldots, task_{n-1}, task_n\}$ and a virtual machine queue. The process of resource allocation is to map the n tasks to the m virtual machine. Assume that $task_i$ is running on vm_j, and the uptime of $task_i$ is expressed as follows:

$$et_{ij} = \frac{taskMI_i}{vmMIPS_j} \tag{1}$$

where $taskMI_i$ is the instruction length of $task_i$ and $vmMIPS_j$ is the number of millions-of-instructions-per-second of vm_j.

According to the actual running time et_{ij} of $task_i$ on vm_j, the direct running cost ec_{ij} of $task_i$ on vm_j is represented as follows:

$$ec_{ij} = \rho \times et_{ij} \tag{2}$$

where ρ is the cost of task running in unit time.

Since multiple tasks may be assigned to the same vm resource, there is a waiting queue for resources and some tasks may wait for some time before running. The waiting time is denoted as follows:

$$wt_{ij} = \sum_{k=1}^{r} et_{kj} \tag{3}$$

where r is the task number of the waiting queue of vm_j before $task_i$ and et_{kj} is the actual running time of $task_k$ on vm_j.

A task may wait some time for vm before running on virtual machine. Therefore, the completion time of task should include waiting time and actual running time, and it is represented as follows:

$$ft_{ij} = et_{ij} + wt_{ij} \tag{4}$$

where et_{ij} is the actual running time of $task_i$ on vm_j and wt_{ij} is the waiting time of $task_i$ for vm_j.

In this work, we proposed a dynamic pricing model to trade cost for time, so the cost that the user needs to pay varies with the waiting time of task in task queue. The cost is negatively related to the waiting time; in other words, the longer the waiting time is, the lower the cost is. However, it is clearly unreasonable that the cost is reduced to be negative as the waiting time is extended, so there must be some constraint between the cost and the waiting time. Moreover, the cost has a certain curve fluctuation in the process of dynamic change. Therefore, we can utilize exponential function to represent the dynamic change of cost and time, and the specific formula is expressed as follows:

$$fc_{ij} = \begin{cases} ec_{ij}, wt_{ij} = 0 \\ ec_{ij} \times \left(1 - e^{-\left(\frac{ft_{ij}}{wt_{ij}} - 1\right)}\right), wt_{ij} \neq 0 \end{cases} \quad (5)$$

where ft_{ij} is the final finish time of $task_i$ on vm_j, wt_{ij} is the waiting time of $task_i$ for vm_j and ec_{ij} is the direct running cost of $task_i$ on vm_j.

In order to ensure the reliability and accuracy of Eq. (5), we give some illustrations on the constraints and characteristics of dynamic pricing model.

- When some task has a high requirement for completion time, the task should be directly allocated to VM after user submits it on cloud testing platform. Therefore, the waiting time of the task is 0, and the corresponding cost equals to the uptime *et*.
- When the waiting time of the task becomes longer, the cost of the task dynamically decreases.
- When the task always remains in the task queue, the waiting time of the task is indefinitely prolonged, the final finish time *ft* is infinitely close to the waiting time *wt* and the cost is infinitely close to 0, but never equals to 0.

The dynamic pricing model can reduce the cost of the user and ease the load of the cloud test platform. However, it still needs a proper scheduling algorithm to ensure that it can meet the user's expectations well.

3.3. Resource scheduling algorithm

User Expectation-based Genetic Algorithm (UEGA for short) is designed to realize the resource scheduling, which can optimize the completion of all tasks in cloud testing platform when meeting the constraints on time and cost.

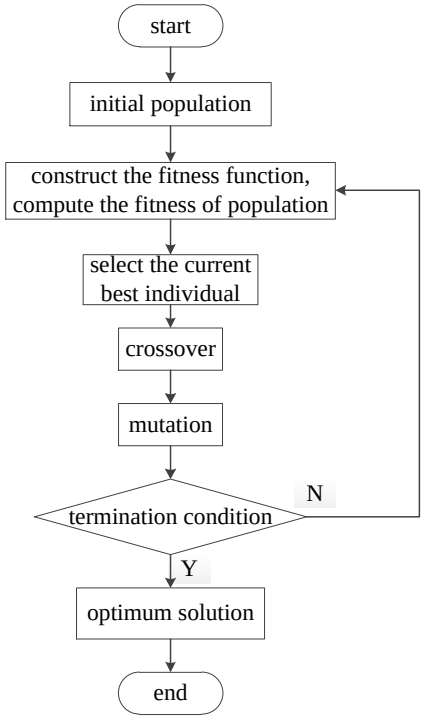

Figure 2. The process of genetic algorithm

Since genetic algorithm is suitable for solving large-scale optimization problems, we use genetic algorithm to realize the task scheduling of cloud testing, and seek the best task allocation scheme to meet the requirements. The concrete scheduling process is shown in Figure 2.

3.3.1. Initial population

The population needs to be initialized before the iterative process of genetic algorithm. The size of the population determines the diversity of the population, while the length of the chromosome is determined by the encoding process. In

this work, we adopt the indirect encoding method with a consistent match between the task and VM resources. Therein, the length of chromosome equals to the number of tasks in task queue while the value of each gene in the chromosome equals to the VM number that is assigned to the task. For example, the value of the *i*th position equals to *j*, which indicates that vm_j is allocated to $task_i$ [6,9].

Suppose that there are 5 VM resources {vm_1, vm_2, ..., vm_5} and 10 tasks {$task_1$, $task_2$, ..., $task_{10}$}, and the corresponding encoding is {2, 1, 3, 4, 1, 3, 5, 2, 4, 1}, which illustrates that the first task runs on the second VM, the second task runs on the first VM, and so on.

Now, we assume that the size of the population is *pop_size*. Besides, the length of chromosome is the length of the task queue. In addition, the value of gene in chromosome equals the VM number. The encoding method as described above is used to initialize the population, and the specific initialization process is shown in Algorithm 1.

Algorithm 1 Initializing population

Input: *pop_size*(the size of population), *chromo_size*(the length of chromosome)
Output: *pop* (initialized population)

```
1.  for i = 1:pop_size
2.    for j=1:chromo_size
3.      chormosome_j←randomized value from 1 to choromo_size  /* Assign values to each locus of the chromosome */
4.    end for
5.    pop_i←chormosome_j
6.  end for
```

3.3.2. Fitness function

Genetic algorithm designs the fitness function to select the next generation of the evolutionary population so as to find the optimal solution of the problem. Fitness function is not only the standard of determining the quality of chromosomes in a population, but also affects the results of task allocation. Therefore, we formulate two constraints based on user expectation and a final goal to select individuals.

Users will present the requirements for time and cost when they submit tasks through the cloud testing platform. In addition, the goal of cloud provider is to meet users' requirement. Therefore, users' expectation should be met as greatly as possible when selecting the population. Namely, the following two constraints are satisfied.

$$\begin{cases} ft_i \le ET_i \\ fc_i \le EC_i \end{cases}, i \in \{1,2,\cdots,n\}$$

where *n* represents the number of tasks in task queue, and also indicates the length of chromosome. ft_i is the completion time of $task_i$ gotten by Eq. (4). And fc_i is the actual cost of $task_i$ gotten by Eq. (5). ET_i is users' time requirement for $task_i$ while EC_i is users' cost requirement for $task_i$.

Users' expectation for time and cost is presented by users. They do not understand the resource scheduling mechanism in cloud testing platform, which may cause all user requirements not to be met when allocating resource to tasks. Therefore, in the process of resource scheduling with genetic algorithm, if there is an individual that can satisfy the two constraints, the fitness value of the individual will be doubled to achieve a preferred selection.

From the point of view of the cloud service provider, in addition to meeting user needs, it is necessary to complete all tasks in the shortest time, which can save energy consumption and improve system efficiency. In fact, in solving the fitness value, at first, the encoding of chromosomes needs to be decoded. In other words, the chromosomes will be decoded into the tasks queue according to the allocated resource. For example, the encoding of chromosomes is {2, 1, 3, 4, 1, 3, 5, 2, 4, 1}, which can be decoded as follows:

$$\begin{cases} vm_1 : \{task_2, task_5, task_{10}\} \\ vm_2 : \{task_1, task_8\} \\ vm_3 : \{task_3, task_6\} \\ vm_4 : \{task_4, task_9\} \\ vm_5 : \{task_7\} \end{cases}$$

Therefore, in order to make the completion time of all tasks shorter, the makespan which means the completion time of the maximum task queue corresponding to the virtual machine resource must be minimized. In addition, an objective function is formulated as follows:

$$f(object) = \min_{s} \max_{j} \sum_{i=1}^{k} ft_{ij} \tag{6}$$

where j represents the serial number of virtual machine, k is the number of task which is processed by vm_j, and s represent the population era number.

The roulette algorithm [1] is used to select the contemporary optimal population. That is to say, the individual with high fitness values is more likely to be selected. However, the target of resource scheduling is to minimize the makespan. Therefore, fitness function is defined as follows:

$$fitness = 1 \Big/ f(object) \tag{7}$$

Therefore, algorithm 2 is designed to describe the process of getting the fitness value of individuals in the population, and the optimal individual is selected according to the fitness distribution of the population.

Algorithm 2 Getting the fitness value

Input: the size of population: *pop_size*, the length of chromosome: *chromo_size*, the number of virtual machine: *res_size*, population distribution: *pop*, the million instruction length of task: *taskMI*, the processing speed of virtual machine: *taskMIPS*;

Output: the fitness value of population: *fitness_value*

```
for  i = 1:pop_size
  for  j=1:chromo_size
    for k=1:res_size
      while(vm_k is allocated to task_j )
        et_jk← taskMI_j/taskMIPS_k
        ec_jk←ρ* et_jk
        rt_size_k++
        if rt_size_k←0
          wt_jk←0
        else
          wt_jk is gotten by Eq.(3)
        end if
        ft_jk is gotten by Eq.(4)
        fc_jk is gotten by Eq.(5)
      end while
    end for
  end for
  totaltime is gotten by Eq.(6)
  fitness_value_i is gotten by Eq.(7)
  while (the cost and time of all tasks within users' expectation)
    fitness_value_i← 2*fitness_value_i
  end while
end for
```

3.3.3. Selection operation

The selection operation is a process of selecting individuals from the previous generation to the next generation according to the distribution of individual fitness. In this work, roulette wheel algorithm [1] is applied to select the best individuals from the contemporary population. Firstly, the fitness value of each individual is obtained according to the decoding rules and the defined fitness function. Then, the individuals are sorted according to the fitness values, and a random number from 0 to the sum of fitness values is generated randomly to select individuals. In fact, the greater the fitness value is, the higher the probability that the individual will be selected is. The probability that each individual is selected is defined as follows:

$$p(j) = \frac{fitness(j)}{\sum_{i=1}^{n} fitness(i)} \tag{8}$$

However, the method has a certain contingency, which may lead to the selection of some ordinary individuals and the elimination of good individuals. Therefore, the elitist scheme is utilized to keep the good individuals to the next generation without crossover and mutation.

3.3.4. Crossover

Crossover is the process of generating two new individuals between two old individuals, thereby enhancing the fitness of chromosomes in populations. General crossover algorithms include single-point crossover and multi-point crossover [6,9]. In this work, we use single-point crossover. At first, a number from 0 to 1 is generated. If the number is less than the crossover rate, the two individuals are crossed. Otherwise, the operation is not performed. Then, a crossover position is needed, and the segments of the chromosome from the crossover position are exchanged between two individuals. The concrete process is described in Figure 3.

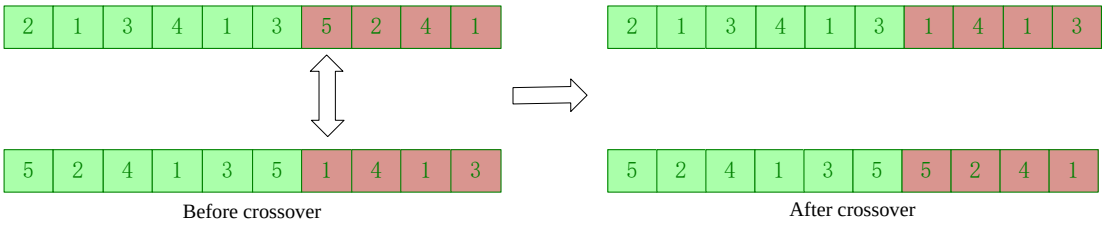

Figure 3. The process of crossover

3.3.5. Mutation

Mutation is the process performed on individuals. At first, a number from 0 to 1 is generated. If the number is less than mutation rate, the individual is mutated. Then, the mutation position needs to be determined. If the position is 0, the mutation is not needed; otherwise, the process is performed as Eq. (9).

$$num_{mut} = (m+1) - num_{cur} \tag{9}$$

where m represents the total number of virtual machine, num_{cur} is the current value of the pending mutation position, and num_{mut} is the value after mutation. The special mutation process is shown in Figure 4.

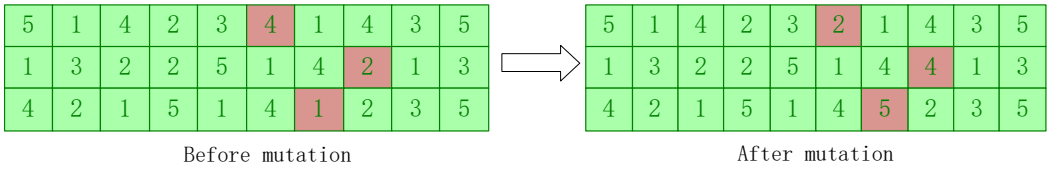

Figure 4. The process of mutation

4. Experimental comparison and analysis

In cloud computing environment, computer resources distributed in different geographical locations can be shared via network, and resource pooling is achieved for users by virtual technology. As a result, the VM resources are allocated when the tasks submitted by users are executed in cloud platform.

In this work, we use CloudSim (Cloud simulator) [2] to carry out the simulation experiments. Function *bindCloudletToVm()* provided in Datacenter is overridden to complete the allocation of virtual machine resources. The Round Robin algorithm (RR for short) and Greedy algorithm are used to compare and verify the feasibility and effectiveness in resource scheduling using user expectation-based genetic algorithm (UEGA for short) proposed in this work. Besides, RR is a resource scheduling algorithm which comes with CloudSim while Greedy algorithm is employed to schedule the resources in ref. [13] which can improve the performance. In addition, T. Wang, et al. [21] proposed *m*-job spanning time and load balancing genetic algorithm (JLGA for short) based on double-fitness adaptive genetic algorithm to

meet users' requirements and improve the resource utilization. Therefore, we choose these three algorithms to compare the results.

In order to verify the feasibility and accuracy of two schemes much better, we construct 50 virtual machines as resources to execute the tasks; they have different processing speed. Besides, 100 tasks with different instruction length need to be processed. Moreover, all these tasks are required to be finished within a certain time and limited cost. Therefore, we simulate users' different expectations on the completion time and cost of different tasks, which act as the constraints when resource scheduling is achieved.

We manage to achieve the optimal resource scheduling using genetic algorithm. The parameters needed for the resource scheduling process of genetic algorithm are shown in Table 1. Therein, *pop_size* represents the size of population, *generation_size* is the number of iteration and acts as the termination condition of iteration, *cross_rate* is the crossover probability and *mutate_rate* denotes the mutation probability.

Table 1 The Parameters of Genetic Algorithm

Parameter	Value
pop_size	200
generation_size	1000
cross_rate	0.9
mutate_rate	0.01

According to the setting parameters, some experiments using four different algorithms are conducted. Therein, JLGA [21] also utilizes genetic algorithm to optimize task scheduling based on some constraints. Therefore, its experimental parameters are the same as ours. Some figures are used to show the experimental performance using four different algorithms.

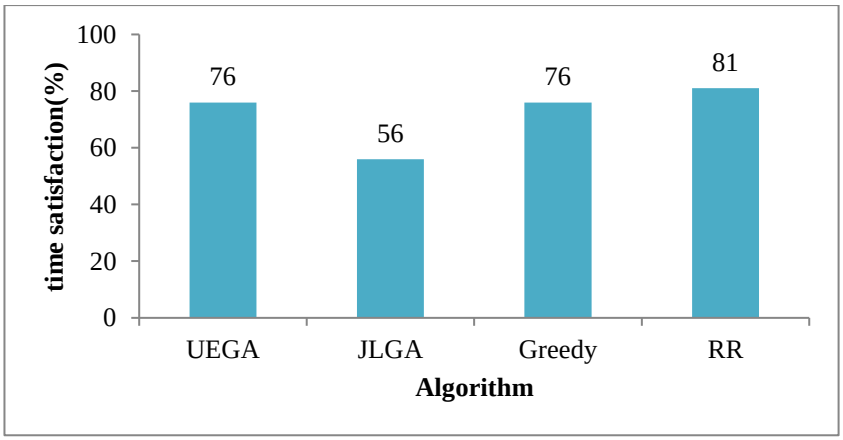

Figure 5. The rate of meeting user satisfaction for completion time using different algorithms

The rate of meeting user satisfaction for completion time using four different algorithms is shown in Figure 5. In other words, the number of tasks whose completion time is within the expected time is demonstrated. Concretely, we can find that the rate of user satisfaction for completion time using the proposed method (UEGA) is 76% while the rate is 56%, 76% and 81% respectively using the other three contrast algorithms. Therein, the rate of time satisfaction using the proposed method is just below that of using RR algorithm. Therefore, it can be concluded that UEGA can meet users' expectation for completion time of tasks commendably.

It is shown in Figure 6 that the rate of meeting user satisfaction for cost is compared using four different algorithms. Specifically, we can find that the rate of user satisfaction for cost is the highest when UEGA is used to achieve task scheduling. Therein, the rate using UEGA is 92% that means the cost of 92 tasks is within the expected cost by users while the other three are respectively 68%, 63% and 49% using the other three algorithms. Therefore, it can be concluded that UEGA can better meet users' expectation for cost of tasks compared with the other three algorithms.

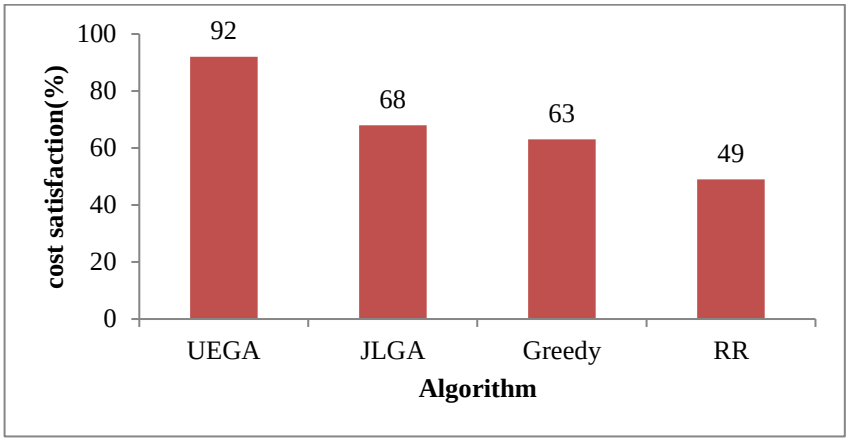

Figure 6. The rate of meeting user satisfaction for cost using different algorithms

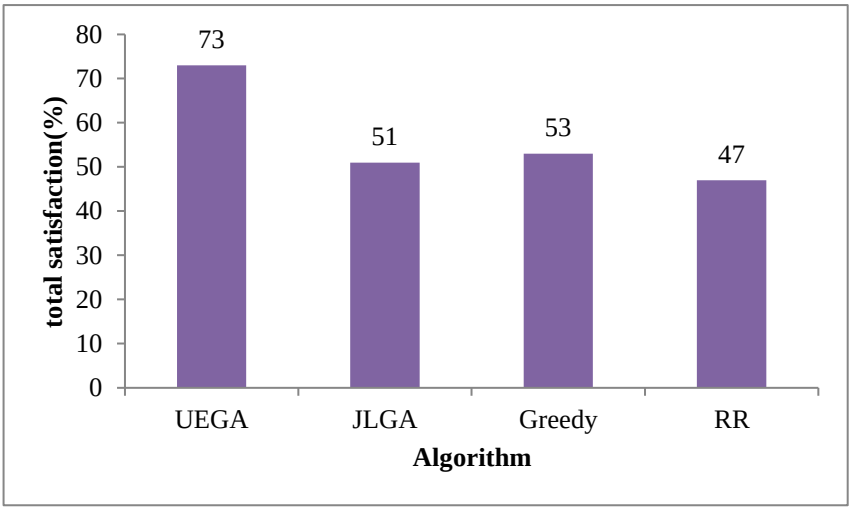

Figure 7. The rate of meeting user satisfaction using different algorithms

Integrating the completion time and cost of each task, the rate of meeting users' total expectation is shown in Figure 7. In other words, the number of the task whose completion time and cost is within users' expectation is demonstrated. It is obvious that the rate of user satisfaction is the highest using the proposed method. Concretely, we can find that the rate of user satisfaction using UEGA is 73% while the other three are respectively 51%, 53% and 47%. Therefore, it can be concluded that UEGA can better meet users' expectation compared with the other three algorithms.

In sum, the resource scheduling based on user expectation is proposed according to users' behaviour characteristics and the properties of cloud testing platform. Through the above analysis of the experimental results, we conclude that UEGA can better meet users' expectation and improve QoS.

Besides, in the premise of meeting user expectation, we evaluate the performance of the proposed algorithm in two other ways. Makespan represents the completion time of last task in task queue, and minimizing the makespan can reduce the energy consumption and improve resource utilization.

The makespan using four different algorithms is shown in Figure 8. We can find that the makespan is 239.91, 334.25, 192.05 and 233.99 when UEGA, JLGA, Greedy and RR are respectively used to achieve the task scheduling. It is obvious that the result using UEGA is lower than that using JLGA, and higher than that using the two others.

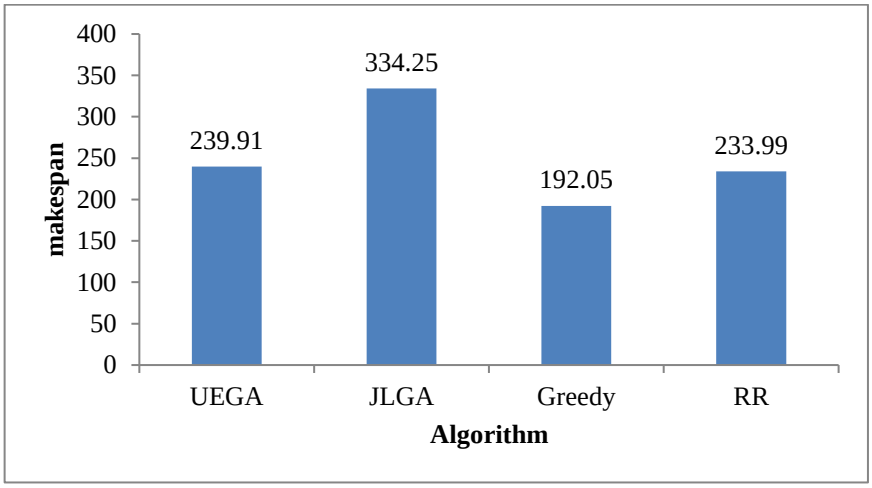

Figure 8. Makespan using different algorithms

Load balancing can effectively ensure the stability of cloud platform. We use the formula proposed in [21] to compute the load, and the specific formula is as follows:

$$load = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(V_j - \overline{V})} \tag{10}$$

where m represents the total number of vm, V_j is the completion time of task queue in vm_j and $\overline{V}$ is the average time of all vm.

The load using four different algorithms is shown in Figure 9. We can find that the load is 60.45, 67.48, 7.26 and 40.21 when UEGA, JLGA, Greedy and RR are respectively used to achieve the task scheduling. It is revealed that the result using UEGA is lower than that using JLGA, and higher than that using the two others.

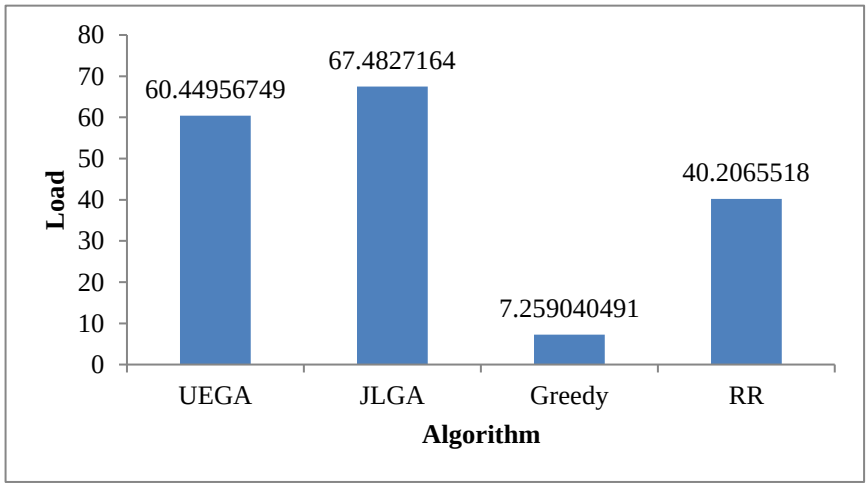

Figure 9. Load using different algorithms

In fact, Greedy algorithm mainly aims at minimizing the completion time of current time, and given experimental data has certain regularity, which is good for RR algorithm. Therefore, the two algorithms perform better than UEGA and JLGA. In addition, the load balancing and spanning time act as fitness function in JLGA. Both makespan and load using UEGA are lower than that using JLGA. Therefore, UEGA also performs well at minimizing the makespan and balancing the load.

All in all, genetic algorithm based on user expectation proposed in this work takes into account the benefits of users and providers of cloud test platform. With the limited resources, UEGA not only meets user expectation and improves QoS, but also comparatively minimizes the makespan and balance load, which serves to save energy consumption and improve the utilization of resources.

5. Conclusions

Cloud testing platform, which has the feature of flexible service, resource pooling, on-demand service, ubiquitous access, and so on, is a new software testing model with Test as a Service (TaaS for short). With the continuous development and improvement of cloud testing platform, the number of users has been sharply increasing. Therefore, how to ensure all users' benefit becomes a challenging issue. Meanwhile, it is necessary to design a good pricing model to ensure the reasonableness of service pricing and propose an effective resource scheduling model to reduce energy consumption and ensure quality of service.

At present, common cloud computing service platforms in the market mostly adopt reservation or lease mechanism to provide the resource and service for users. However, this method may lead to wasting resource because there may be some users who do not need such resources during the lease. Therefore, some platforms provide on-demand service mechanism to reduce the waste of resource. However, it requires users to pay more for each task. As a result, a dynamic pricing model is designed to flexibly convert between time and cost in this work, which can ensure that users requirement for every task is comprehensively considered. In addition, user expectation-based genetic algorithm is proposed to schedule the resources in cloud testing, which can balance the benefits between users and cloud providers. In other words, the benefit of cloud platform is maximized in the premise of meeting user demand. In this work, users' demand for time and cost of each task is treated as a constraint while minimizing the makespan is treated as the goal. In order to meet the constraint and achieve the goal, the following work is done in cloud test resource scheduling.

- In order to meet the users' different demands for cost and time of tasks and ensure the quality of service as well as possible, the concept of user expectation is introduced to represent the different demands for time and cost of tasks submitted by different users.
- A dynamic pricing model, which uses trade-cost-for-time mechanism, is designed by analyzing the different expectations of the user community and considering the characteristics of cloud platform. In the dynamic pricing model, there is a negative correlation between cost and waiting time of tasks, which can better satisfy different kinds of needs.
- Genetic algorithm is applied to achieve the optimal resource scheduling and minimize the makespan.
- Resource scheduling scheme based on user expectation in cloud testing, which comprehensively considers the benefits of users and cloud providers, is proposed to reduce the energy consumption and balance the load in the premise of ensuring the quality of service.

To conclude, the experimental results show the proposed resource scheduling scheme based on user expectation in cloud testing can greatly guarantee the need for task completion time and cost of different users and reduce the completion time of all tasks.

Future studies in this area could shed light on how to better improve the performance of cloud testing platform. In other words, balancing the load and reducing the energy consumption are considered in detail in the premise of minimizing the makespan. In addition, a behaviour prediction model will be designed to foresee user needs for completion time and cost of tasks according to users' behavioural characteristics, which can better ensure the quality of service.

Acknowledgements

Thanks go first to the anonymous referees for their sound comments and suggestions. This work is partly supported by the National Natural Science Foundation of China under Grant Nos. 61762041 and 61462030, and the Science and Technology Project of Jiangxi Provincial Department of Education of China under Grant No. GJJ160427.

References

1. I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "GA-ETI: An Enhanced Genetic Algorithm for the Scheduling of Scientific Workflows in Cloud Environments," *Journal of Computational Science*, Available online 4 September 2016 (https://doi.org/10.1016/j.jocs.2016.08.007)
2. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software Practice & Experience*, vol. 41, no. 1, pp. 23-50, August 2011
3. S. Chaisiri, B. S. Lee, and D. Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164-177, February 2011
4. D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The Analytic Hierarchy Process: Task Scheduling and Resource Allocation in Cloud Computing Environment," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 1-14, June 2013
5. J. Gao, X. Bai, W. T. Tsai, and T. Uehara, "Testing as a Service (TaaS) on Clouds," in *IEEE Seventh International Symposium*

on Service-Oriented System Engineering, pp. 212-223, Redwood, USA, March 2013
6. H. Hallawi, J. Mehnen, and H. He, "Multi-Capacity Combinatorial Ordering GA in Application to Cloud Resources Allocation and Efficient Virtual Machines Consolidation," *Future Generation Computer Systems*, vol. 69, pp. 1-10, November 2016
7. J. Huang, "The Workflow Task Scheduling Algorithm Based on the GA Model in the Cloud Computing Environment," *Journal of Software*, vol. 9, no. 4, pp. 873-880, April 2014
8. J. Huang, R. J. Kauffman, and D. Ma, "Pricing Strategy for Cloud Computing: A Damaged Services Perspective, " *Decision Support Systems*, vol. 78, pp. 80-92, November 2015
9. D. Jung, T. Suh, H. Yu, and J. M. Gil, "A Workflow Scheduling Technique Using Genetic Algorithm in Spot Instance-Based Cloud," *Ksii Transactions on Internet & Information Systems*, vol. 8, no. 9, pp. 3126-3145, September 2014
10. A. K. Kar and A. Rakshit, "Flexible Pricing Models for Cloud Computing Based on Group Decision Making Under Consensus," *Global Journal of Flexible Systems Management*, vol. 16, no. 2, pp. 191-204, June 2015
11. A. V. Katherine and D. K. Alagarsamy, "Conventional Software Testing Vs. Cloud Testing," *International Journal of Scientific & Engineering Research*, vol. 3, no. 9, pp. 1-5, September 2012
12. R. Kumar and S. Singh, "Cloud Testing: Perspective and Challenges," *International Journal of Computer Applications*, vol. 106, no. 17, pp. 975-8887, November 2014
13. P. Liu, "Cloud Computing 3th ed.," Publishing House of Electronics Industry, Beijing, China, August 2015(in Chinese)
14. P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Communications of the ACM*, vol. 53, no. 6, pp. 50-50, June 2010
15. B. Narula and V. Beniwal, "Cloud Testing- Types, Service Platforms and Advantages," *International Journal of Computer Applications*, vol. 72, no. 20, pp. 1-6, June 2013
16. C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the Optimal Allocation of Virtual Resources in Cloud Computing Networks," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1060-1071, February 2013
17. B. Qiao, R. Cai, D. Chen, H. Wang, Y. Chen, and G. Wang, "Resource Scheduling Optimization Algorithm for Xen Virtual Machines, " *Journal of Software*, vol. 25, no. S2, pp. 201-202, December 2014 (in Chinese with English abstract)
18. X. Shi and K. Xu, "Utility Maximization Model of Virtual Machine Scheduling in Cloud Environment," *Chinese Journal of Computers*, vol. 36, no. 2, pp. 252-262, February 2013(in Chinese with English abstract)
19. G. Tian, D. Meng, and J. Zhang, "Reliable Resource Provision Policy for Cloud Computing," *Chinese Journal of Computers*, vol. 33, no. 10, pp. 1859-1872, October 2010 (in Chinese with English abstract)
20. J. T. Tsai, J. C. Fang, and J. H. Chou, "Optimized Task Scheduling and Resource Allocation on Cloud Computing Environment Using Improved Differential Evolution Algorithm," *Computers & Operations Research*, vol. 40, no. 12, pp. 3045-3055, December 2013
21. T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai, "Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing," in *IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 146-152, Chengdu, China, July 2014
22. X. Wu, J Hou, S. Zhuo, and W. Zhang. "Dynamic Pricing Strategy for Cloud Computing with Data Mining Method," *Communications in Computer & Information Science*, vol. 207, pp. 40-54, January 2013
23. W. Wei, Y. Liu, and W. Yang, "A Fast Approximation Algorithm for the General Resource Placement Problem in Cloud Computing Platform," *Journal of Computer Research and Development*, vol. 53, no. 3, pp. 697-703, March 2016 (in Chinese with English abstract)
24. Z. Xiao, W. Song, and Q. Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment, " *IEEE Transactions on Parallel & Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, June 2013
25. P. Xiao and Z. Tang, "Game Theory-based Resource Pricing Model in Cloud Platforms," *International Journal of Communication Networks & Distributed Systems*, vol. 14, no. 3, pp. 256-271, January 2015
26. Y. Yuan, C. Wang, C. Wang, T. Ren, and B. Liu, "An Uncompleted Information Game Based Resources Allocation Model for Cloud Computing," *Journal of Computer Research and Development*, vol. 53, no. 6, pp. 1342-1351, June 2016 (in Chinese with English abstract)
27. X. Zhao, B. Zhang, and C.Zhang, "Service Selection Based Resource Allocation for SBS in Cloud Environments," *Journal of Software*, vol. 26, no. 4, pp. 867-885, April 2015 (in Chinese with English abstract)
28. Y. Zheng, L. Cai, S. Huang, and Z. Wang, "VM Scheduling Strategies Based on Artificial Intelligence in Cloud Testing," in *IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 1-7, Las Vegas, NV, USA, June 2014
29. Y. Zheng, L. Cai, S. Huang, J. Lu, and P. Liu, "Cloud Testing Scheduling Based on Improved ACO," in *International Symposium on Computers and Informatics*, pp. 569-578, Istanbul, Turkey, January 2015.

Zhongsheng Qian graduated from the School of Computer Engineering and Science, Shanghai University, China, for the degree of Ph. D. He entered the Post-doctoral station of Computer Science & Technology Department in Jiangxi University of Finance & Economics, China, from 2013 to 2015. He visited Aalborg University, Denmark as a guest professor in 2016. Now he is an associate professor of the School of Information Technology, Jiangxi University of Finance & Economics, Nanchang, China. His research interests include cloud testing, Web testing, software engineering, formal verification.

Xiaojin Wang is a master student from the School of Information Technology, Jiangxi University of Finance & Economics, Nanchang, China. His research interests include cloud testing, software modelling & testing, Web testing.