IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 1, 2020, accepted August 15, 2020, date of publication August 18, 2020, date of current version August 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017643

PSO-GA-Based Resource Allocation Strategy for Cloud-Based Software Services With Workload-Time Windows

ZHEYI CHEN[1], LIJIAN YANG[2], YINHAO HUANG[2], XING CHEN[2], XIANGHAN ZHENG[2], AND CHUNMING RONG[3], (Senior Member, IEEE)

[1]Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter EX4 4QF, U.K.
[2]College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China
[3]Department of Electronic Engineering and Computer Science, University of Stavanger, 4036 Stavanger, Norway

Corresponding authors: Xing Chen (chenxing@fzu.edu.cn) and Xianghan Zheng (xianghan.zheng@fzu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004800, and in part by the Talent Program of Fujian Province for Distinguished Young Scholars in Higher Education, and the China Scholarship Council.

ABSTRACT Cloud-based software services necessitate adaptive resource allocation with the promise of dynamic resource adjustment for guaranteeing the Quality-of-Service (QoS) and reducing resource costs. However, it is challenging to achieve adaptive resource allocation for software services in complex cloud environments with dynamic workloads. To address this essential problem, we propose an adaptive resource allocation strategy for cloud-based software services with workload-time windows. Based on the QoS prediction, the proposed strategy first brings the current and future workloads into the process of calculating resource allocation plans. Next, the particle swarm optimization and genetic algorithm (PSO-GA) is proposed to make runtime decisions for exploring the objective resource allocation plan. Using the RUBiS benchmark, the extensive simulation experiments are conducted to validate the effectiveness of the proposed strategy on improving the performance of resource allocation for cloud-based software services. The simulation results show that the proposed strategy can obtain a better trade-off between the QoS and resource costs than two classic resource allocation methods.

INDEX TERMS Cloud-based software services, resource allocation, QoS prediction, workload-time windows, particle swarm optimization, genetic algorithm.

I. INTRODUCTION

In cloud computing, different resources including central processing units (CPUs), memories, and storage units in data centers are virtualized to provide users with renting services, which is expected to promise the on-demand resource provisioning and authorize users the basic access to massive data and cloud resources [1]–[3]. However, changeable workloads commonly occur in cloud environments, which may seriously degrade the QoS when resource demands rapidly increase [4]. In order to guarantee the scalability and resiliency of cloud resources, cloud service providers (CSPs) need to offer on-demand configurations of software and hardware resources in the shared infrastructure [5].

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

In recent years, widespread applications of cloud-based software services have proved that cloud software engineering is gaining a strong development momentum [6]. However, it is still challenging to balance the QoS and resource costs when allocating resources for cloud-based software services under such dynamic cloud environments [7]. In response to this challenge, it is necessary to develop an adaptive technology to improve the effectiveness of resource allocation for cloud-based software services.

A. LITERATURE REVIEW

Although there are classic methods that can address the resource allocation problem to some extent, most of them allocate resources based on the current workload in cloud environments. With the consideration of the current workload, AlQayedi *et al.* [8] proposed a queuing-theory based scheme to estimate the number of virtual machines (VMs)

needed for meeting the requirements of response time. In this scheme, the CPU utilization was regarded as a threshold, the workload status was checked regularly (e.g., every 1 or 2 minutes), and then the number of VMs to be adjusted (i.e., add or delete) was calculated. However, it is hard for this scheme to produce effective resource allocation plans when future workloads fluctuate significantly. Maurer *et al.* [9] designed a resource-efficient decision-making method in response to workload fluctuations, where a rule-based knowledge management strategy was proposed to achieve the autonomous reconstruction of VMs. Zahid *et al*. [10] designed a ruled-based language for CSPs with adaptation schemes, in order to enhance the QoS compliance in high-performance computing (HPC) clouds. However, these two works did not regard the minimization of resource costs as an optimization goal, which may lead to excessive resource costs. Ficco *et al*. [11] proposed a meta-heuristic resource allocation approach with the game theory for optimizing policies. Moreover, a clustering-based heuristic approach for edge resource allocation was proposed in [12] to reduce the average service response time of applications. However, these two works only considered the deterministic user demands without implementing the dynamic resource allocation. When the workloads changed, it may not well meet the QoS requirements. Using the control theory, Haratian *et al*. [13] designed a resource allocation framework for guaranteeing the QoS, where the fuzzy controller was used to determine the resource provisioning in the control loop. Moreover, Avgeris *et al*. [14] developed a resource allocation mechanism for computation offloading with the control theory, which can reduce the response time and resource costs when processing application tasks. However, these two traditional control-theoretic methods need massive iterations to search for better resource allocation plans, which may cause excessive resource waste if VMs are frequently discontinued. Based on the PSO algorithm, Xie *et al.* [15] developed a strategy of resource allocation with the price adjustment, where a utility function was designed to evaluate the QoS. According to the user demands from workloads, the resource agent can dynamically adjust the resource price, in order to obtain the maximum profit for each workload. However, the PSO algorithm may fall into the local convergence, and thus it is hard to obtain the overall optimal solution of resource allocation. Moreover, Kumar and Gondhi [16] designed a self-scaling model that can dynamically adjust resource provisioning referring to the QoS indicators, which performed the resource correction at the VM level while considering the situations of under-utilization and over-utilization for resources. Although this model can make on-demand resource adjustments for the applications under changeable workloads, it did not well consider the relationship between the QoS and VM rental costs nor take into account future workload changes with complex fluctuations. Besides, Chen *et al.* [17] proposed an adaptive resource management framework for cloud-based software services. This framework first trained the QoS prediction model by using historical data. Next, it utilized the PSO-based algorithm to determine future resource allocation plans with the predicted QoS. Finally, the feedback control was introduced to achieve the desired effect of resource allocation. However, this work did not consider the impact of future workloads when making the QoS prediction, which may seriously degrade the model effectiveness and limit its performance for cloud resource allocation.

B. CONTRIBUTIONS

When dealing with the resource allocation problem for cloud-based software services, the workload status has become one of the most essential factors that affect the performance of resource allocation [18]. For example, when many users make requests at the same time, workloads burst and thus the available resources might be inadequate for satisfying the requirements of cloud-based software services. On the contrary, when workloads remain low, the excessive allocation may lead to resource waste. Therefore, the workload prediction can be regarded as a feasible solution for promising efficient resource allocation in cloud environments [19]–[21]. If the cloud resource allocation plans are developed only based on the current workload, the effectiveness of resource allocation will undoubtedly be affected by future workload changes. However, it is difficult to exactly match the predicted workloads with the actual results. Meanwhile, the system performance in the scheduling stage may be seriously affected by the inaccurate workload prediction [22]–[24]. To this end, some advanced methods, e.g., [4] and [25], to cite two of the most recent, have been proposed to provide the high accuracy of workload prediction, and thus the scheduling efficiency can be improved. Next, we assume that future workloads have been precisely predicted with the support of the advanced method. Furthermore, the predicted workloads in workload-time windows are utilized to better handle the problem of cloud resource allocation. With the consideration of the above problems, inspired by the work in [26], we propose an adaptive resource allocation strategy for cloud-based software services with workload-time windows. The main contributions of this article are summarized as follows.

- The problem of resource allocation for cloud-based software services is formulated by introducing workload-time windows. Further, we build a calculation model for optimizing VM resource allocation plans.
- A new PSO-GA based decision-making method is proposed with the combination of the QoS prediction model. Specifically, the PSO-GA has absorbed the advantages of the PSO and the GA and improved their inadequacies in population diversity, search range, and convergence speed. Moreover, the current and future workloads in workload-time windows are utilized during the process of calculating resource allocation plans. Furthermore, through integrating the PSO-GA with workload-time windows, the proposed method can effectively adjust the current resource allocation plan to approach the objective one.

TABLE 1. Main symbols used in the problem formulation.

Symbol	Description
W	Various workloads in different time intervals
w_i	Workload in a time interval
ADD	Various operations of adding VM resources at different times
add_i	Operation of adding VM resources at the time t_i
VM	Allocated VM resources in different time intervals
QoS_i	QoS value at the time t_i
$Cost_i$	Costs of renting VMs at the time t_i

- The extensive simulation experiments using the RUBiS benchmark are conducted to demonstrate the effectiveness of the proposed resource allocation strategy for cloud-based software services. The results show that the proposed strategy can make a better trade-off between the QoS and resource costs compared to the other two classic resource allocation methods.

The rest of this article is organized as follows. In Section II, we formulate the problem of resource allocation for cloud-based software services. Section III discusses the proposed resource allocation strategy in detail. In Section IV, we show the performance evaluation for the proposed strategy with the RUBiS benchmark. Section V concludes this article and looks for future work.

II. PROBLEM FORMULATION

In this section, we formulate the problem of resource allocation for cloud-based software services by introducing workload-time windows.

The dynamic changes of cloud environments may result in different QoS for cloud-based software services, and these changes can be divided into external and internal environmental changes. More specifically, the external environmental changes are commonly caused by external factors (e.g., dynamic workloads), and the internal ones usually are affected by cloud management systems (e.g. different types and numbers of VMs with various rental prices in resource allocation plans). With the consideration of the above two factors, the main symbols used in the problem formulation are described in Table 1.

As mentioned before, the external factors refer to dynamic workloads. Therefore, we assume that the workload changes are defined by the following piecewise function, where each workload is constant during a specific time interval.

$$W = \begin{cases} w_0, & t_0 \le t < t_1 \\ w_1, & t_1 \le t < t_2 \\ \ldots, & \ldots \\ w_i, & t_i \le t < t_{i+1} \\ \ldots, & t \ge t_{i+1}, \end{cases} \tag{1}$$

where each workload is defined as a 2-tuple, denoted by $w_i = \langle n_i, r_i \rangle$, with the same workload duration. n_i is the number of requests and r_i is the read-write rate of requests at time t_i, respectively. When $t \ge t_{i+1}$, workloads cannot be observed.

Moreover, the above-mentioned internal factors refer to different types and numbers of VMs with various rental prices in resource allocation plans. Because VMs are commonly charged by hours when they are leased, we assume that each VM is leased for one hour each time and it will be automatically shut down after one hour. Therefore, we only need to consider the number of VMs to be added when adjusting resource allocation plans. For various workloads in different time intervals, the corresponding operations of adding VM resources are defined as

$$ADD = \begin{cases} add_0, & t = t_0 \\ add_1, & t = t_1 \\ \ldots, & \ldots \\ add_i, & t = t_i \\ \ldots, & t \ge t_{i+1}. \end{cases} \tag{2}$$

In each resource allocation plan, there are m types of VMs, denoted by $Type = \langle 1, 2, 3, \ldots, m \rangle$, that are available to be added. Therefore, the operation of adding VM resources at the time t_i can be defined as

$$add_i = \{a_i^1, a_i^2, \ldots, a_i^j, \ldots, a_i^m\}, \tag{3}$$

where a_i^j indicates the number of VMs of the j-th type to be added at the time t_i.

Therefore, the allocated VM resources in different time intervals can be described as

$$VM = \begin{cases} add_0, & t_0 \le t < t_1 \\ \ldots, & \ldots \\ \sum_{k=max\{0,i-q+1\}}^{i} add_k, & t_i \le t < t_{i+1} \\ \ldots, & t \ge t_{i+1}, \end{cases} \tag{4}$$

where $q = 1h/\Delta t$ is the maximum number of unexpired VMs and $\Delta t = t_{i+1} - t_i$ indicates the duration of each workload. In each time interval, the allocated VM resources are calculated by adding up the unexpired VMs.

When allocating resources to cloud-based software services, a common objective of cloud engineers or self-adaptive systems is to balance the relationship between the QoS and resource costs, which can be expressed by using an objective function [27].

On one hand, QoS_i represents the QoS value at the time t_i, which is usually specified by the service level agreements (SLAs) with corresponding performance indicators, including the response time (RT) and data throughput (DT) [28]. Specifically, the RT indicates the time that a user needs to wait for a response when requesting a service, and the DT represents the amount of information that a system can process in a given time. However, these two indicators cannot be used to predict the QoS values of a system because they can be monitored only after completing the resource allocation. To this end, it is necessary to design an effective QoS prediction model [29]–[31]. Based on our previous work in [32], the QoS prediction model can be defined as

$$QoS_{predicted} = QoS(w, vm), \tag{5}$$

where the input of the model includes workloads and the corresponding allocated VM resources, and the output of the model is the predicted QoS values.

On the other hand, $Cost_i$ represents the costs of renting VMs (i.e., the operation of adding VMs) at the time t_i. We denote the rental prices of each type of VMs as $P = \langle p_1, p_2, p_3, \ldots, p_m \rangle$, and thus $Cost_i$ can be defined as

$$Cost_i = \sum_{j=1}^{m} a_i^j \times p_j. \tag{6}$$

Therefore, the objective function can be described as

$$Minimize \ \ w_1 \times \frac{1}{QoS_i} + w_2 \times Cost_i, \tag{7}$$

where w_1 and w_2 are used to weight $\frac{1}{QoS_i}$ and $Cost_i$, and these two weights are defined by cloud engineers based on their experience.

However, it is hard to well guarantee the effectiveness of resource allocation plans if they are developed only based on the current workload. Through introducing workload-time windows, the operations of adding VM resources can be taken according to both current and future workloads, and thus the resource allocation plans in different time intervals can be calculated more effectively. This is because the workloads within a period of future time can be observed by using workload-time windows when allocating resources at the current moment. Thus, the current resource allocation plan can be accordingly adjusted and become more effective.

We assume that the workload-time window i can be used to predict the future workloads with the length of l (i.e., the number of time intervals), and thus the corresponding workload W^i can be defined as

$$W^i = \begin{cases} w_i, & t_i \le t < t_{i+1} \\ w_{i+1}, & t_{i+1} \le t < t_{i+2} \\ \ldots, & \ldots \\ w_{i+l-1}, & t_{i+l-1} \le t < t_{i+l}. \end{cases} \tag{8}$$

Moreover, the operations of adding VM resources within the window i can be defined as

$$ADD^i = \begin{cases} add_i, & t = t_i \\ add_{i+1}, & t = t_{i+1} \\ \ldots, & \ldots \\ add_{i+l-1}, & t = t_{i+l-1}. \end{cases} \tag{9}$$

Correspondingly, the allocated VM resources within the window i can be defined as

$$VM^i = \begin{cases} \sum_{k=max\{0,i-q+1\}}^{i} add_k, & t_i \le t < t_{i+1} \\ \ldots, & \ldots \\ \sum_{k=max\{0,i-q+j-1\}}^{i+j} add_k, & t_{i+j} \le t < t_{i+j+1} \\ \ldots, & \ldots \\ \sum_{k=max\{0,i-q+l\}}^{i+l-1} add_k, & t_{i+l-1} \le t < t_{i+l}. \end{cases} \tag{10}$$

Therefore, the problem is transferred to explore the objective resource allocation plan by adjusting VM resources in workload-time windows, which can be regarded as an NP-hard combinatorial optimization problem.

III. WORKLOAD-TIME WINDOWS ENABLED ADAPTIVE RESOURCE ALLOCATION

A. FRAMEWORK OVERVIEW

In this section, we present the proposed adaptive resource allocation strategy for cloud-based software services with workload-time windows in detail. Figure 1 illustrates an overall framework of the proposed strategy, where the main steps are described as follows.

Step 1: Initialize the data and parameters of workload-time windows, including the workloads and corresponding allocated resources in different time intervals.

Step 2: Utilize the PSO-GA with the support of the QoS prediction model to search for the objective resource allocation plan in workload-time windows.

Step 3: Make adjustments to the current resource allocation plan to approach the objective one.

B. PSO-GA FOR OBJECTIVE RESOURCE ALLOCATION

1) HYBRID OF PSO AND GA

The particle swarm optimization (PSO) simulates the bird migration in the nature [33], which continuously makes particles iterating to explore the optimal solution. As an important element in the PSO, a particle represents a candidate solution for an optimization problem. In the solution space, particles are updated iteratively by comparing the optimal values in the individual and population histories. In the PSO, the velocity and position equations of a particle, denoted by V_i^{t+1} and X_i^{t+1}, are defined as

$$V_i^{t+1} = wV_i^t + c_1 r_1 (pBest_i^t - X_i^t) + c_2 r_2 (gBest_i^t - X_i^t), \tag{11}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}, \tag{12}$$

where V_i^t and X_i^t are the velocity and position of the i-th particle at the t-th iteration, respectively. w is used to weight the ability of particles in keeping the velocity. $pBest_i^t$ and $gBest^t$ are the optimal values in the individual and population histories at the t-th iteration, respectively. r_1 and r_2 represents random factors. c_1 and c_2 are learning factors that control the learning ability of particles. Moreover, a fitness function is used to evaluate particles.

The genetic algorithm (GA) is used to simulate the evolutionary process in organisms by using computational models [34]. Following the principle of the fittest survival in nature, the GA initially uses random solutions and then searches for better ones. During this process, the crossover and mutation operations are taken to the excellent individuals in the previous generation, and thus the next generation with better performance can be produced.

Through combining the advantages of the PSO and GA, we propose a particle swarm optimization and genetic algorithm (PSO-GA) based resource allocation method for cloud-based software services. With the support of the PSO, the proposed method optimizes the individual particles in

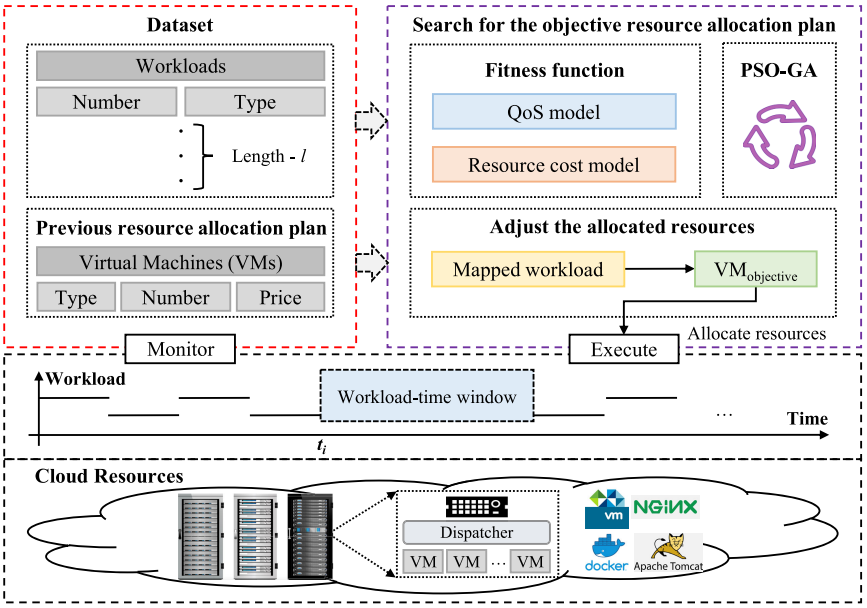

FIGURE 1. Overview of the proposed cloud resource allocation strategy.

the GA, and thus the problem of low efficiency in the later-stage searching of the GA can be well addressed. More specifically, the PSO-GA first sorts the particles based on their evaluation values of the fitness function, where the high-performance individuals will be preserved in the next iteration and the poor-performance ones will be eliminated. Next, the crossover and mutation operations will be taken on these high-performance individuals and the remaining particles will go to the next generation. In this article, the particles represent potential resource allocation plans within a workload-time window. After a fixed time interval, the PSO-GA will be used to select an optimal particle as the objective resource allocation plan at the current moment. The following will provide detailed descriptions of the particle coding, fitness function, and update strategy in the PSO-GA.

2) PARTICLE CODING

As for the PSO-GA based resource allocation for cloud-based software services, the discrete coding [33] is used to encode particles. In the workload-time window k with the length of l, ADD^k (defined in Equation (9)) is regarded as a particle X, where the particle represents a resource allocation plan within the window k. When there are m types of VMs, the i-th particle at the t-th iteration can be defined as

$$X_i^t = (x_{i11}^t, \ldots, x_{il1}^t, \ldots, x_{ilm}^t, \ldots, x_{ilm}^t), \tag{13}$$

where the particle X_i^t is composed of $m \times l$ elements. Each element indicates the number of different types of VMs to be added in the workload-time window.

Figure 2 shows an example of the particle coding, where the length of the window is 3 and the number of VM types is 3. This example indicates that the different types of VMs added in the 1st, 2nd and 3rd resource allocation plan are (1, 2, 0), (1, 0, 3) and (2, 1, 0), respectively.

FIGURE 2. An example of the particle coding.

3) FITNESS FUNCTION

In workload-time windows, resource allocation plans are evaluated by the fitness function. The fitness function commonly makes a trade-off between the QoS and resource costs, which can be used to guide the PSO-GA in the right direction for finding the optimal solution. With the consideration of workload-time windows and the optimization objective in Equation (7), the fitness function can be defined as

$$\begin{aligned} Fitness^i &= w_1 \times \frac{1}{QoS_i} + w_2 \times Cost_i \\ &= w_1 \times \sum_{k=i}^{i+l-1} \frac{1}{QoS(w_k, vm_k)} \\ &\quad + w_2 \times \sum_{k=i}^{i+l-1} \sum_{j=1}^{m} a_k^j \times p_j, \end{aligned} \tag{14}$$

where $w_k \in W^i$ and $vm_k \in VM^i$. The total QoS values in the window can be calculated by using Equation (5). Moreover, a_k^j is the number of VMs of the j-th type to be added, the p_j indicates the rental costs of the j-th type of VMs. Thus, the total resource costs in the window can be calculated. The above two parts make up the fitness function, and it can be noticed that better resource allocation plans (with higher QoS and lower resource costs) in a workload-time window are with smaller values of the fitness function.

4) UPDATE STRATEGY OF PARTICLES

In the PSO-GA, the state of the particle swarm is updated by using the mutation and crossover operations.

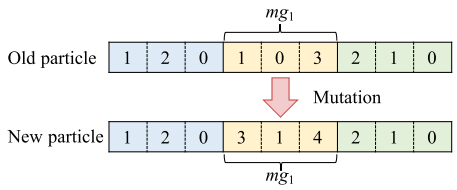

FIGURE 3. Mutation in the particle update.

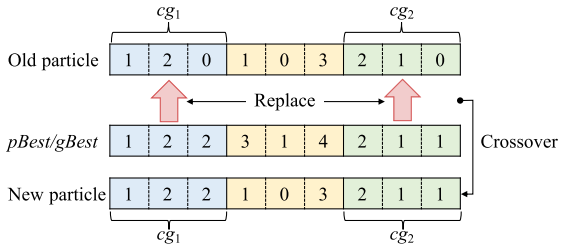

FIGURE 4. Crossover in the particle update.

On one hand, the mutation operation randomly selects a gene segment in the particle and irregularly mutates the gene values within a specific threshold. For example, Figure 3 illustrates the mutation operation for the particle encoded in Figure 2, where a gene segment, denoted by mg_1, is randomly selected and the corresponding gene values are mutated from ((1, 0, 3) to (3, 1, 4).

On the other hand, Figure 4 illustrates an instance of the crossover operation in the particle update. More specifically, two gene segments, denoted by cg_1 and cg_2, are first randomly selected. Next, the corresponding gene values are replaced by the local or global optimum (denoted by *pBest* and *gBest*, respectively) with the same crossover probability.

5) PROCESS OF RESOURCE ALLOCATION

As shown in Algorithm 1, the main steps of the PSO-GA for searching the objective resource allocation plan are described as follows.

Step 1: (Lines 2∼6) Initialize the parameters, including the population size N, the maximum number of iterations, add_i, *pBest*, and *gBest*.

Step 2: (Lines 7∼18) When the execution conditions are satisfied, the particle swarm is first updated by taking the crossover and mutation operations. Next, the particle i is evaluated, and *pBest* and *gBest* are updated by comparing the values of the fitness function.

Step 3: (Line 19) Finish iterations and output *gBest*.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed PSO-GA based resource allocation strategy for cloud-based software services and make comparisons with the other two classic methods.

A. DATASETS AND SETTINGS

The simulation environment is established based on the CloudStack [35], where the Kernel-based Virtual Machine (KVM) is used as the hypervisor that is run on a physical server with Intel® Core^{TM} i7-6700HQ CPU @2.60 GHZ, RAM 8.00 GB, and CentOS 6.2 64-bit. As shown in Table 2, there are three types (i.e., small, medium, and large) of VMs with different rental prices. Furthermore, the RUBiS benchmark [36], a prototype of the auction site that simulates user behaviors under different workload patterns on the *eBay.com*, is used to evaluate the proposed strategy. Through running the RUBiS benchmark on the CloudStack with a period of one month, we collect the runtime datasets used for experiments, which include the current and objective resource allocation plans with different QoS under various workloads. Moreover, we assume that the workload (i.e., the number of clients) is distributed in [2000, 7000] with two types of tasks including reading (e.g., browsing) and writing (e.g., bidding and rating). Besides, the proposed strategy is implemented based on Python 3.6, where the essential parameters including the population size, the number of iterations, and the mutation rate are set to 50, 100 and 0.5, respectively.

As shown in Figure 5, a Sigmoid function is used to map the response time (RT) to the interval [0, 1] according to the empirical data, which reflects the QoS values.

The simulation experiments are conducted under the premise of continuous workloads for a known time period. Specifically, we define a workload-time window as

Algorithm 1 PSO-GA for Searching the Objective Resource Allocation Plan

Input: Population size N.
Output: Global optimal solution *gBest*.

1: **Procedure**
2: **for** each particle i **do**
3: Initialize add_i for the particle i.
4: Evaluate the particle i and set $pBest_i = add_i$.
5: **end for**
6: $gBest = min\{pBest_i\}$.
7: **while** not stop **do**
8: **for** $i = 1$ **to** N **do**
9: Update the particle i by mutate and crossover.
10: Evaluate the particle i.
11: **if** $Fitness^i(add_i) < Fitness^i(pBest_i)$ **then**
12: $pBest_i = add_i$.
13: **end if**
14: **if** $Fitness^i(pBest_i) < Fitness^i(gBest)$ **then**
15: $gBest = pBest_i$.
16: **end if**
17: **end for**
18: **end while**
19: Output *gBest*.
20: **End Procedure**

TABLE 2. Different types of VMs with various rental prices.

Property	Small	Medium	Large
CPU core	1	1	1
Memory (GB)	1	2	4
Price (RMB/hour)	1.761	1.885	2.084

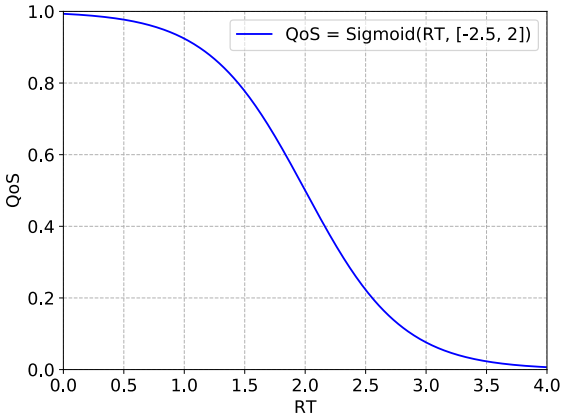

FIGURE 5. Mapping from RT to QoS.

TABLE 3. Workload changes with different read-write ratios.

Time Interval (min)	Workload	Reading	Writing
0∼30	2500	0.55	0.45
30∼60	3000	0.45	0.55
60∼90	4000	0.30	0.70
90∼120	6000	0.50	0.50
120∼150	6500	0.80	0.20
150∼180	6000	0.05	0.95
180∼210	3500	0.65	0.35
210∼240	4500	0.25	0.75
240∼270	2000	0.40	0.60
270∼300	2500	0.75	0.25
300∼330	3000	0.50	0.50
330∼360	2500	0.40	0.60
360∼390	3000	0.50	0.50
390∼420	3000	0.90	0.10
420∼450	3000	0.40	0.60
450∼480	3000	0.45	0.55
480∼510	3000	0.45	0.55
510∼540	3000	0.45	0.55

$W^i = \{w_i, w_{i+1}, w_{i+2}\}$ with the length of 90 minutes. Therefore, the workload is predicted every 30 minutes. Table 3 shows the workload changes within the predicted 540 minutes. To guarantee the high prediction accuracy, we assume that the workloads after 540 minutes cannot be observed. Moreover, the workloads in different time intervals are with corresponding read-write ratios.

Moreover, better resource allocation plans are common with smaller values of the fitness function. The pre-defined weights w_1 and w_2 in Equation (14) reflect different preferences of cloud engineers for the QoS and resource costs. In practical cloud systems, the common objective of the fitness function is to balance the QoS and resource costs. However, it is hard to achieve this objective due to the complex relationship between the QoS and resource costs in cloud-based software services. To relieve this problem, we set $w_1 = 900$ and $w_2 = 1/6$ according to our experience. Specifically, the following fitness function is used in our simulation experiments.

$$Fitness^i = 900 \times \sum_{k=i}^{i+2} \frac{1}{QoS_k} + \frac{1}{6} \times \sum_{k=i}^{i+2} Cost_k. \quad (15)$$

To validate the effectiveness of the proposed strategy, comparative experiments are conducted with the other two baseline methods including the greedy algorithm and the single-point optimal local random method [37], which are commonly used in the problem of cloud resource allocation. The detailed settings are described as follows.

- Greedy algorithm. The algorithm does not consider the workload changes in workload-time windows. Moreover, the optimal resource configuration at the last moment will be used to calculate the optimal resource allocation plan under the current workload.
- Single-point optimal local random method. For each workload within the same time window, the method first traverses all resource allocation plans to find a plan with the smallest value of the fitness function, which is called the single-point optimal plan. Next, two VMs will be randomly added or subtracted based on this single-point optimal plan, and a feasible resource allocation plan within the window will be obtained under the same running time (i.e., 2 minutes) as the PSO-GA.
- PSO-GA. In each iteration, the evaluation value of each particle will be first calculated by using the fitness function and the global and local optimal particles will be selected. Next, these particles will be updated by the mutation and crossover operations and the iterations continue.

B. EXPERIMENTAL RESULTS

Based on the above settings of simulation experiments, we evaluate the performance of the proposed PSO-GA based resource allocation strategy for cloud-based software services and make comparisons with the greedy algorithm and single-point optimal local random method.

Figures 6 to 8 show the resource allocation plans generated by using the greedy algorithm, single-point optimal local random method, and the proposed PSO-GA strategy under various workloads in different time intervals, respectively. As we can see from the results, the VM resources can be roughly adjusted by all these three methods according to the workload changes at different times. When the workloads are high, more VMs tend to be leased in the corresponding resource allocation plans. Otherwise, fewer VMs would be used. However, various types and numbers of VMs may be adopted in the same time interval by using these three methods, which would result in diverse QoS and resource costs. To further analyze the effectiveness of our proposed resource allocation strategy, we regard the QoS values, resource costs, and the total values of the fitness function (obtained by using Equation (15)) as performance indicators and evaluate the above three resource allocation methods in the following simulation experiments.

As shown in Figure 9, the average values of the fitness function of each time interval within the time period from 0 to 390 minutes are compared among different methods. It can be noticed that the proposed PSO-GA strategy outperforms both the greedy algorithm and the single-point optimal

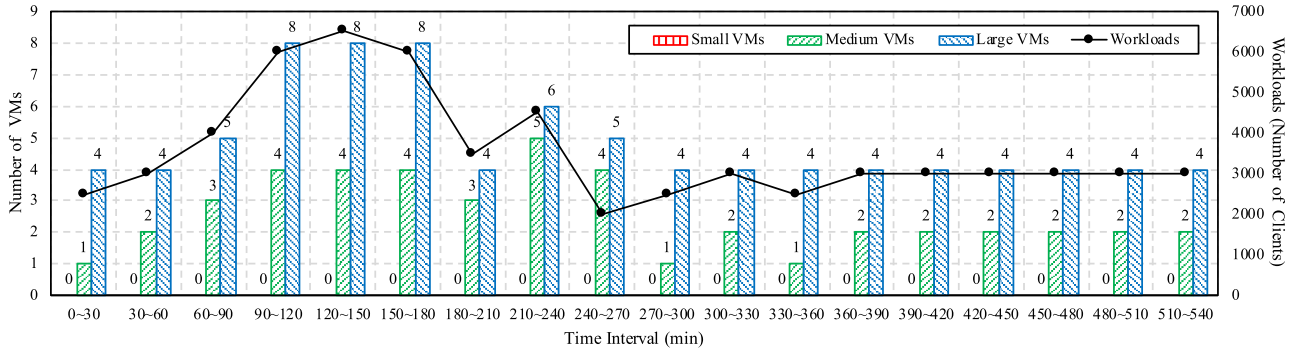

FIGURE 6. Resource allocation plans generated by using the greedy algorithm under various workloads in different time intervals.

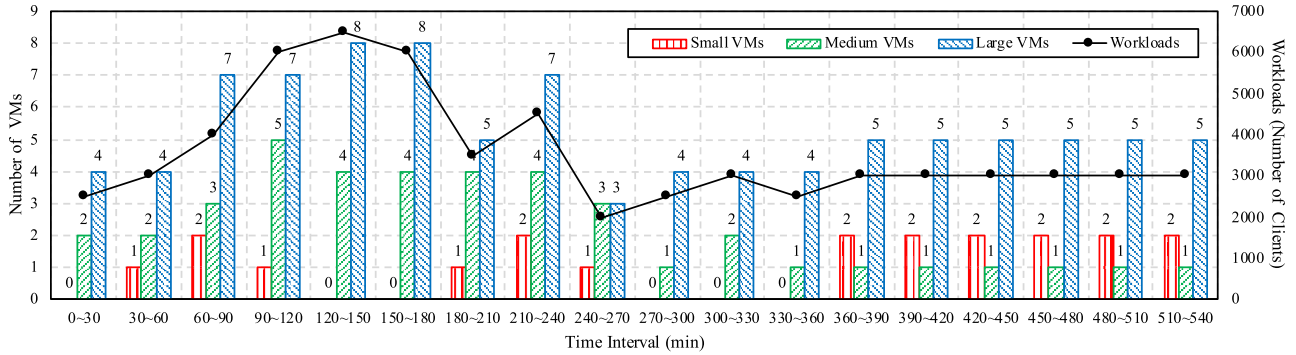

FIGURE 7. Resource allocation plans generated by using the single-point optimal local random method under various workloads in different time intervals.

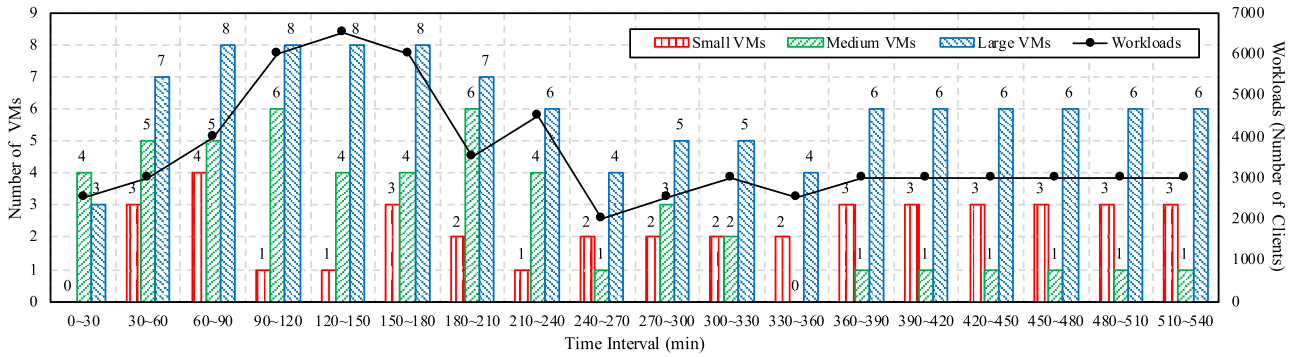

FIGURE 8. Resource allocation plans generated by using the proposed strategy under various workloads in different time intervals.

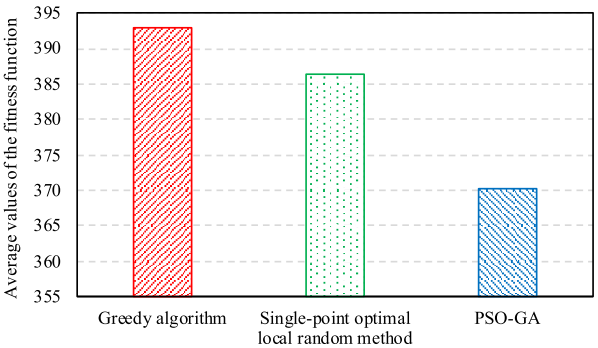

FIGURE 9. Comparisons of the average values of the fitness function among different methods.

local random method with the improvement of over 5% and 4%, respectively. This is because the PSO-GA used in the proposed strategy not only pays attention to the evolution process between each generation of the population but also attaches importance to the retention and re-maturation of excellent individuals, which can thus enhance the population diversity and make the potential resource allocation plans closer to the objective one. By contrast, it is difficult for the single-point optimal local random method to get close to the objective resource allocation plan because the search style adopted by this method is random without a clear target. Moreover, the greedy algorithm only considers the optimal resource allocation plan at the current workload, and thus the plans may fluctuate with workload changes and it cannot stably obtain the global optimal solution.

Furthermore, we evaluate the performance of different resource allocation methods in terms of QoS and resource costs, respectively. As shown in Figure 10, on one hand,

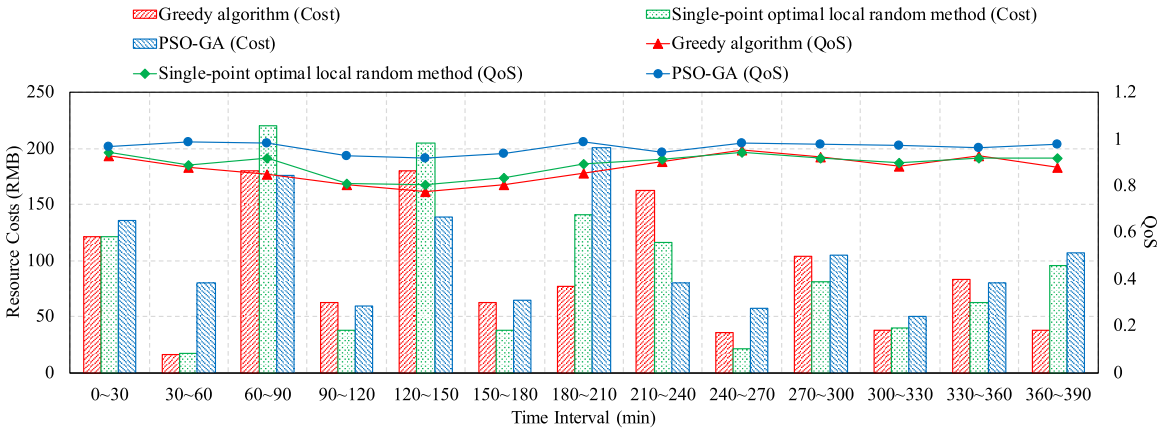

FIGURE 10. Comparisons of the resource costs and QoS among different methods.

TABLE 4. Influence of the running time of the proposed PSO-GA on the average values of the fitness function.

Running time (min)	Average values of the fitness function
1.0	375.62
1.5	372.79
2.0	371.32
2.5	370.94
3.0	370.75

the proposed PSO-GA strategy can maintain stable QoS values (between 0.91∼0.99) within the whole time period (0∼390 minutes) with the average of around 0.96, which outperforms the other two methods. This is because the proposed strategy considers future workloads during resource allocation but the other two methods do not take workload-time windows into account. Even when the workloads rise rapidly between the time period of 60∼150 minutes, the proposed strategy can also keep the QoS at a high level. By contrast, the QoS values fluctuate seriously when using the greedy algorithm. Moreover, the QoS values locate in 0.80∼0.94 when the single-point optimal local random method is used. On the other hand, we compare the resource costs of the proposed strategy with the other two methods. The results show that the average resource costs of using the greedy algorithm and the single-point optimal local random method in each time interval are around 89.24 RMB and 91.94 RMB, respectively. Although the proposed strategy leads to slightly higher average resource costs, they are still within an acceptable range under the premise of ensuring the QoS.

From the above simulation experiments, we can find that the proposed strategy outperforms the single-point optimal local random method within the same running time. Although the running speed of the greedy algorithm is fastest among the three methods, it cannot well guarantee good performance indicators. Furthermore, we evaluate and analyze the influence of the running time of the proposed PSO-GA on the performance of resource allocation in terms of the average values of the fitness function in different time intervals. As shown in Table 4, although the running time affects the performance of the PSO-GA to some extents, the PSO-GA can still keep stable and high performance on resource allocation (low values of the fitness function) even if the running time of the algorithm is less than 2 minutes (e.g., 1 or 1.5 minutes). From the perspective of system management, this optimization time delay is acceptable.

V. CONCLUSION AND FUTURE WORK

In dynamic cloud environments with fluctuating workloads, it is hard to guarantee the effectiveness of resource allocation plans if they are developed and performed based on the current workload. To address this problem, in this article, we propose an adaptive resource allocation strategy for cloud-based software services by introducing workload-time windows. The proposed strategy takes the current and future workloads into the process of producing resource allocation plans, where the PSO-GA based runtime decision-making method is utilized to explore the objective resource allocation plan. The extensive simulation experiments are conducted to verify the effectiveness of the proposed strategy. The results show that the proposed strategy outperforms the other two classic resource allocation methods and it can achieve a better trade-off between the QoS and resource costs.

In real-world cloud environments, workloads are usually continuous and thus it has become a new challenge to make use of such continuous workloads in workload-time windows during the decision-making process of cloud resource allocation. In the future, we will continue our work by using the learning-based algorithms (e.g., reinforcement learning) and introduce a continuous workload function to further enhance the performance of resource allocation under dynamic cloud environments with continuous workloads.

ACKNOWLEDGMENT

(Zheyi Chen and Lijian Yang contributed equally to this work.)

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] V. Prokhorenko and M. A. Li Babar, "Architectural resilience in cloud, fog and edge systems: A survey," *IEEE Access*, vol. 8, pp. 28078–28095, 2020.

[3] M. Ala'Anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access*, vol. 7, pp. 141868–141887, 2019.

[4] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[5] X. Chen, F. Zhu, Z. Chen, G. Min, X. Zheng, and C. Rong, "Resource allocation for cloud-based software services using prediction-enabled feedback control with reinforcement learning," *IEEE Trans. Cloud Comput.*, early access, May 4, 2020, doi: 10.1109/TCC.2020.2992537.

[6] W. A. Simm, F. Samreen, R. Bassett, M. A. Ferrario, G. Blair, J. Whittle, W. A. Simm, and P. Young, "SE in ES: Opportunities for software engineering and cloud computing in environmental science," in *Proc. 40th IEEE/ACM Int. Conf. Softw. Eng., Softw. Eng. Soc. (ICSE-SEIS)*, May/Jun. 2018, pp. 61–70.

[7] T. Chen and R. Bahsoon, "Self-adaptive and online QoS modeling for cloud-based software services," *IEEE Trans. Softw. Eng.*, vol. 43, no. 5, pp. 453–475, May 2017.

[8] F. AlQayedi, K. Salah, and M. J. Zemerly, "Adaptive cloud resource allocation scheme to minimize SLO response time violation," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2016, pp. 1–5.

[9] M. Maurer, I. Brandic, and R. Sakellariou, "Self-adaptive and resource-efficient SLA enactment for cloud computing infrastructures," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 368–375.

[10] F. Zahid, A. Taherkordi, E. G. Gran, T. Skeie, and B. D. Johnsen, "A self-adaptive network for HPC clouds: Architecture, framework, and implementation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 12, pp. 2658–2671, Dec. 2018.

[11] M. Ficco, C. Esposito, F. Palmieri, and A. Castiglione, "A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation," *Future Gener. Comput. Syst.*, vol. 78, pp. 343–352, Jan. 2018.

[12] L. Zhao, J. Wang, J. Liu, and N. Kato, "Optimal edge resource allocation in IoT-based smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 30–35, Mar. 2019.

[13] P. Haratian, F. Safi-Esfahani, L. Salimian, and A. Nabiollahi, "An adaptive and fuzzy resource management approach in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 907–920, Oct. 2019.

[14] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, "Adaptive resource allocation for computation offloading: A control-theoretic approach," *ACM Trans. Internet Technol.*, vol. 19, no. 2, p. 23, 2019.

[15] F. Xie, Y. Du, and H. Tian, "A resource allocation strategy based on particle swarm algorithm in cloud computing environment," in *Proc. 4th Int. Conf. Digit. Manuf. Automat.*, Jun. 2013, pp. 69–72.

[16] D. Kumar and N. K. Gondhi, "A QoS-based reactive auto scaler for cloud environment," in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst. (ICNGCIS)*, Dec. 2017, pp. 19–23.

[17] X. Chen, H. Wang, Y. Ma, X. Zheng, and L. Guo, "Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model," *Future Gener. Comput. Syst.*, vol. 105, pp. 287–296, Apr. 2020.

[18] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 1, pp. 90–100, Mar. 2014.

[19] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct./Dec. 2015.

[20] J. Kumar and A. K. Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution," *Future Gener. Comput. Syst.*, vol. 81, pp. 41–52, Apr. 2018.

[21] X. Tang, "Large-scale computing systems workload prediction using parallel improved LSTM neural network," *IEEE Access*, vol. 7, pp. 40525–40533, 2019.

[22] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, and V. C. M. Leung, "Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1688–1699, Jun. 2018.

[23] T. Ahammad, U. K. Acharjee, and M. M. Hasan, "Energy-effective service-oriented cloud resource allocation model based on workload prediction," in *Proc. 21st Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2018, pp. 1–6.

[24] H. M. Nguyen, G. Kalra, T. J. Jun, S. Woo, and D. Kim, "ESNemble: An echo state network-based ensemble for workload prediction and resource allocation of Web applications in the cloud," *J. Supercomput.*, vol. 75, no. 10, pp. 6303–6323, Oct. 2019.

[25] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with CloudInsight for predictive resource management," *IEEE Trans. Cloud Comput.*, early access, May 27, 2020, doi: 10.1109/TCC.2020.2998017.

[26] H. Wang, Y. Ma, X. Zheng, X. Chen, and L. Guo, "Self-adaptive resource management framework for software services in cloud," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. With Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 1528–1529.

[27] A. Soltanian, D. Naboulsi, R. Glitho, and H. Elbiaze, "Resource allocation mechanism for media handling services in cloud multimedia conferencing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1167–1181, May 2019.

[28] I. V. Paputungan, A. F. M. Hani, M. F. Hassan, and V. S. Asirvadam, "Real-time and proactive SLA renegotiation for a cloud-based system," *IEEE Syst. J.*, vol. 13, no. 1, pp. 400–411, Mar. 2019.

[29] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, Oct. 2017.

[30] W. Hussain and O. Sohaib, "Analysing cloud QoS prediction approaches and its control parameters: Considering overall accuracy and freshness of a dataset," *IEEE Access*, vol. 7, pp. 82649–82671, 2019.

[31] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 134–144, May 2019.

[32] X. Chen, J. Lin, B. Lin, T. Xiang, Y. Zhang, and G. Huang, "Self-learning and self-adaptive resource allocation for cloud-based software services," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 23, p. e4463, 2019.

[33] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.

[34] O. Kramer, *Genetic Algorithm Essentials*. Springer, 2017.

[35] N. Sabharwal, *Apache CloudStack Cloud Computing*. Birmingham, U.K.: Packt, 2013.

[36] *RUBiS: Rice University Bidding System Benchmark*. Accessed: Jan. 2, 2020. [Online]. Available: http://rubis.ow2.org/

[37] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

ZHEYI CHEN received the B.Sc. degree from Shanxi University, China, in 2014, and the M.Sc. degree from Tsinghua University, China, in 2017, in computer science. He is currently pursuing the Ph.D. degree in computer science with the University of Exeter, U.K. His research interests include cloud computing, mobile edge computing, deep learning, reinforcement learning, and resource optimization.

LIJIAN YANG received the B.S. degree in computer science from Fujian Normal University, Fujian, China, in 2019. He is currently pursuing the M.S. degree in computer technology with the College of Mathematics and Computer Science, Fuzhou University. His current research interests include system software, edge computing, and cloud computing.

YINHAO HUANG received the B.S. degree in software engineering from Fuzhou University, Fujian, China, in 2018, where he is currently pursuing the M.S. degree in technology of computer application with the College of Mathematics and Computer Science. Since September 2018, he has also been a student of the Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University. His current research interests include deep neural network, edge computing, and cloud computing.

XING CHEN received the B.S. and Ph.D. degrees in computer software and theory from Peking University, Beijing, China, in 2008 and 2013, respectively. He is currently an Associate Professor and the Deputy Director of the Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, and also leads the Systems Research Group. He has authored or coauthored more than 30 journal and conference articles. His research interests include the software systems and engineering approaches for cloud and mobility. His current projects cover the topics from self-adaptive software, computation offloading, and model-driven approach. He was a recipient of the first Provincial Scientific and Technological Progress Award, in 2018.

XIANGHAN ZHENG received the M.Sc. degree in distributed system and the Ph.D. degree in information communication technology from the University of Agder, Norway, in 2007 and 2011, respectively. He is currently a Professor with the College of Mathematics and Computer Sciences, Fuzhou University, China. His current research interests include new generation network with a special focus on cloud computing services and applications, and big data processing and security.

CHUNMING RONG (Senior Member, IEEE) is currently a Professor and the Head of the Center for IP-based Service Innovation (CIPSI), University of Stavanger (UiS), Norway. He is also the Chair of the IEEE Cloud Computing and an Executive Member of Technical Consortium on High-Performance Computing (TCHPC) and the Chair of STC on Blockchain in the IEEE Computer Society, and has served as the Global Co-Chair of the IEEE Blockchain, in 2018. He is also an Advisor of the StandICT.EU to support European scandalization activities in ICT. He is also a Co-Founder of two start-ups bitYoga and Dataunitor in Norway, both received EU Seal of Excellence Award in 2018. He was an Adjunct Senior Scientist leading Big-Data Initiative at NORCE from 2016 to 2019 and the Vice President of CSA Norway Chapter from 2016 to 2017. His research work focuses on cloud computing, data analytics, cyber security, and blockchain.

Prof. Rong has been honoured as a member of the Norwegian Academy of Technological Sciences (NTVA), since 2011. He has extensive contact network and projects in both the industry and academic. He has served as the Steering Chair from 2016 to 2019, a Steering Member and an Associate Editor of the IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC) since 2016. He is also a Founder and the Steering Chair of the IEEE CloudCom conference and workshop series. He is the co-Editors-in-Chief of the *Journal of Cloud Computing* (Springer) (ISSN: 2192-113X). He has supervised 26 Ph.D. students, nine Postdoctoral researchers, and more than 60 master's projects. He has extensive experience in managing large-scale Research and Development projects, both in Norway and EU.