Chen *et al. J Wireless Com Network* (2021) 2021:24
https://doi.org/10.1186/s13638-021-01912-8

EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

A proactive resource allocation method based on adaptive prediction of resource requests in cloud computing

Jing Chen*, Yinglong Wang and Tao Liu

*Correspondence: jingchen94@163.com
Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250101, China

Abstract
With the development of big data and artificial intelligence, cloud resource requests present more complex features, such as being sudden, arriving in batches and being diverse, which cause the resource allocation to lag far behind the resource requests and an unbalanced resource utilization that wastes resources. To solve this issue, this paper proposes a proactive resource allocation method based on the adaptive prediction of the resource requests in cloud computing. Specifically, this method first proposes an adaptive prediction method based on the runs test that improves the prediction accuracy of resource requests, and then, it builds a multiobjective resource allocation optimization model, which alleviates the latency of the resource allocation and balances the utilizations of the different types of resources of a physical machine. Furthermore, a multiobjective evolutionary algorithm, the Nondominated Sorting Genetic Algorithm with the Elite Strategy (NSGA-II), is improved to further reduce the resource allocation time by accelerating the solution speed of the multiobjective optimization model. The experimental results show that this method realizes the balanced utilization between the CPU and memory resources and reduces the resource allocation time by at least 43% (10 threads) compared with the Improved Strength Pareto Evolutionary algorithm (SPEA2) and NSGA-II methods.

Keywords: Cloud computing, Adaptive short-term prediction, Proactive resource allocation, Balanced resource utilization, Multiobjective optimization

1 Introduction

Cloud computing provides massive computing, storage and network resources to support various services. Users can not only obtain various resources and services anytime and anywhere but also scale out or in resources to ensure their applications' performance or low cost. With the development of big data and artificial intelligence, cloud resource requests possess the characteristics of being diverse, coming in bursts and being sudden. The existing cloud resource allocation methods cannot guarantee the timeliness and optimization of the resource allocation for a large number of sudden resource requests. However, users pay more attention to the timeliness and optimization of their emergent resource requests that can guarantee their applications' performance, and cloud service providers are highly concerned with how to manage the massive resources and improve the resource utilization. An efficient resource allocation method is crucial to meet these goals.

The resource allocation process is a problem to find the suitable physical servers upon which to place virtual machines (VMs). In previous studies, some simple heuristic algorithms, such as round robin (RR) [1], best fit (BF) [2] and min–max [3], are applied to solve the cloud resource allocation problem for small-scale cloud platforms. These algorithms are simple and easy, but they are also prone to wasting resources, especially in large-scale cloud platforms. The bin packing problem is one classical cloud resource allocation method. The VM placement problem on physical servers is transformed into the bin packing problem of packing n objects into m boxes, which requires that all objects be placed in the minimum number of boxes. A dynamic bin packing method is proposed to reduce the total cost of cloud resource allocation by permanently closing empty boxes [4]. Another classical cloud resource allocation method is to model VM placement as a mathematical multiobjective optimization problem. The main idea is to express a cloud resource allocation problem as a multiobjective mathematical function and then to use a multiobjective evolutionary algorithm to solve it.

Although these methods are effective, most of them do not rapidly allocate resources for a large number of sudden resource requests and reduce resource waste of servers. An effective cloud resource allocation method should meet the following conditions for such requests. First, a large number of sudden resource requests should be processed in a timely manner. Second, the optimal resources should be provided to satisfy the resource requests. Third, the physical servers should be used as little as possible, and the proportion between the different types of resources (number of CPU cores, memory capacity and disk size) should be as uniform as possible to reduce resource waste.

The contributions of this paper are summarized as follows.

(1) We propose a runs test (RT)-based adaptive prediction algorithm for resource requests. This algorithm is built based on our previously studied ensemble empirical mode decomposition (EEMD)-Autoregressive Integrated Moving Average model (ARIMA) and EEMD-RT-ARIMA algorithms [5, 6], and it can select a more accurate algorithm to implement the short-term prediction of resource requests via an adaptive prediction strategy.

(2) We propose a proactive resource allocation strategy that combines the active prediction and the passive response of resource requests, which can allocate resources in advance for the future sudden resource requests to guarantee the timelessness of the resource allocation.

(3) We further propose a resource proportion matching model to ensure the uniform usage of different types of server resources, which can reduce resource waste. Then, a mathematical multiobjective optimization problem of the resource allocation is formulated.

(4) We improve the Nondominated Sorting Genetic Algorithm with the Elite Strategy (NSGA-II) to accelerate the solution speed of the multiobjective optimization mathematical problem, which further ensures the timelessness of the resource allocation.

The rest of this paper is organized as follows. Section II introduces related works. Section III describes the proactive resource allocation approach. Section IV presents the experiments and analysis. Section V concludes this paper.

A list of the mathematical notations used in this paper is given in Table 1.

2 Related works

There are some cloud resource allocation methods for big data applications [7], cloud-based software services [8], scientific applications [9], cloud manufacturing [10], workflows [11] and cloud healthcare [12]. Some algorithms or mechanisms have been applied in resource allocation, such as the grasshopper optimization algorithm (GOA) [13], ant-colony optimization and deep reinforcement learning [14], the data-driven probabilistic model [15], and auction mechanisms [16]. Some researchers have proposed the existing resource and task scheduling methods. A systematic review classifies task scheduling approaches into the single cloud environment, multicloud environments and mobile cloud environments for different aims [17]. A comprehensive survey divides the scheduling techniques into three categories: heuristic, meta-heuristic and hybrid schemes [18]. Recently, state-of-the-art multiobjective VM placement mechanisms have been introduced [19]. A review of auction-based resource allocation mechanisms has been comprehensively conducted [20]. Resource allocation methods involve in the aims of reducing cost, minimizing the energy consumption, improving the resource utilization and guaranteeing the quality of service (QoS). A performance-cost grey wolf optimization (PCGWO) algorithm has been proposed to reduce the processing time and cost of tasks [21]. A JAYA algorithm has been used to optimize VM placement and minimize the energy consumption [22]. A fair resource allocation method has been proposed to rapidly and fairly allocate resources and maximize the resource utilization via a flow control policy [23]. These methods cannot provide an effective mechanism to ensure the timelessness of the resource allocation for a large number of sudden resource requests. A multidimensional resource allocation model MDCRA that uses a single weight algorithm (SWA) and a double weight algorithm (DWA) to minimize the number of physical servers, save energy and maximize the resource utilization in cloud computing has been proposed [24]. It models the multidimensional resource allocation problem as a vector bin packing problem. The bin packing problem is an NP hard problem. At present, there is no polynomial complexity optimization algorithm to solve it. Moreover, it only solves the resource capacity constraint without considering incompatible constraints.

Table 1 List of mathematical notations

Symbol	Annotation	Symbol	Annotation
T_{th}	The predefined threshold of main types of VMs	$Q1$	First quartile
$Q3$	Third quartile	IQR	Interquartile range
$M(t)$	Current number of VM requests at t time	$D(t+h)$	The predicted number of VM requests at $t+h$ time
$N(t)$	The total number of VM requests at t time	x_{ij}	The mapping element between a VM and a PM
MD_{ij}	Resource performance matching distance	MPM_{ij}	resource proportion matching distance

An energy-efficient resource allocation scheme considers the energy consumption of the CPU and RAM to reduce the overall energy costs and maintain the service level agreement (SLA) [25]. An empirical adaptive cloud resource provisioning model has been proposed to reduce the latency of the resource allocation and SLA violations via speculative analysis [26]. Both methods focus on workload consolidation and prediction with one target of reducing SLA violations while our method considers the trade-off among the number of physical machines, resource performance and proportional matching. A levy-based particle swarm optimization algorithm has been proposed to minimize the number of running physical servers and balance the load of physical servers by reducing the particle dispersion loss [27]. A dynamic resource allocation algorithm has been proposed to solve resource scheduling and resource matching problems [28]. Here, the Tabu search algorithm is used to solve the resource scheduling problem, the weighted bipartite graph is used to solve the resource matching problem for the tasks on the edge servers, and an optimal solution is further proposed to schedule resources between the edge servers and a cloud data center. This algorithm concentrates on the resource scheduling between the edge server and cloud, but our method focuses on VM placement in a cloud data center. In addition, a cloud workflow scheduling algorithm is proposed based on an attack-defense game model, where a task-VM mapping algorithm is presented to improve the workflow efficiency and different VMs are provided for workflow executions [29]. A fog computing trust management approach that assesses and manages the trust levels of the nodes is proposed to reduce the malicious attacks and the service response time in the fog computing environment [30]. Everything-as-a-Resource as a paradigm is proposed to design collaborative applications for the web [31].

The proactive resource allocation method based on prediction is an effective solution to ensure the timelessness of resource allocation. One type of prediction method is based on machine learning. A prediction-based dynamic multiobjective evolutionary algorithm, called NN-DNSGA-II [32], has been proposed by combining an artificial neural network with the NSGA-II [33]. This algorithm first uses the neural network to predict the pareto-optimal solutions as the initial population of the NSGA-II and then solves the multiobjective optimization problem. The empirical results demonstrate that this algorithm outperforms nonprediction-based algorithms in most cases for the Pegasus workflow management system. However, this algorithm cannot predict the future VM requests, but it does predict a better solution to solve the workflow scheduling problem, which cannot alleviate the resource allocation delay for the future VM request increases. A hybrid wavelet neural network method has been proposed to improve the prediction accuracy through training the wavelet neural network with two heuristic algorithms [34]. The machine learning-based prediction needs to conduct training using a large amount of data, which increases the time consumption and thus cannot guarantee a timely resource allocation. A generic algorithm (GA)-based prediction method has been proposed, and its prediction accuracy is better than the gray model at improving the resource utilization of VMs and physical machines (PMs) [35]. An anti-correlated VM placement algorithm, in which the VMs and the overloaded hosts are predicted to provide the suitable VM placement, has been proposed to reduce the energy consumption [36]. Another type of prediction method is based on statistics. ARIMA is a classical prediction model for time series, and it can also be combined with other methods to predict nonstationary time series. ARIMA and other methods are often combined to implement prediction. A model combining the ARIMA and fuzzy regression, in which the prediction accuracy is improved by setting sliding windows, has been proposed to predict network traffic [37]. An adaptive workload forecasting method dynamically selects the best method from the simple exponential smoothing (SES), ARIMA and linear regression (LR) methods to improve the workload forecasting accuracy [38]. However, this method uses the previous predictions of a set of models and different amounts of training data to execute the next prediction, which increases the prediction cost. A framework combining the ARIMA and LR methods has been used to predict VM and PM workloads, PM power consumption and their total costs [39]. The combination of the ARIMA and Back Propagation Neural Network (BPNN) methods improves the workload prediction accuracy and promotes the minimization of the cost of an edge cloud cluster [40]. An adaptive prediction model has been used to select the best one from the LR, ARIMA and support vector regression (SVR) methods to obtain better prediction results according to workload features [41]. An ensemble model ESNemble combines five different prediction algorithms and extracts their features to forecast the workload time series based on an echo state network, and it outperforms each single algorithm in terms of the prediction accuracy [42]. The above methods combine the classic ARIMA model with other prediction methods, which improve the prediction accuracy to a certain degree. However, these methods may achieve low prediction accuracy for the current resource request sequences with complex characteristics and strong fluctuations. Data preprocessing should be performed to smooth the extremely nonstationary sequences to enhance the prediction accuracy. Our previously proposed EEMD-ARIMA and EEMD-RT-ARIMA algorithms improve the prediction accuracy through decomposing a nonstationary sequence into a few relatively stationary component sequences via EEMD method [5, 6]. The main difference between EEMD-RT-ARIMA and EEMD-ARIMA methods is that EEMD-RT-ARIMA method reduces the cumulative error and the prediction time by selecting and reconstructing the component sequences with similar characteristics into few component sequences based on RT values when the original sequence has weak fluctuation. RT [43] is a method to check the randomness of a sequence. A RT is defined a component with successive symbols (0 or 1). For instance, a sequence '111,001,110,011'' includes three components with successive "1" and two components with successive "0." Each component with successive "1" or "0" is regarded as a RT. The total RT number reflects the random fluctuation of the sequence. Any time series can be changed into a sequence with successive symbols (0 or 1) [44]. The larger the total number of RT, the stronger the sequence fluctuates. Once the original sequence has strong fluctuation determined by the RT, EEMD-ARIMA method can get higher prediction accuracy than EEMD-RT-ARIMA due to more stationary component sequences.

3 Methods

To address the resource allocation lagging behind resource requests, we propose a proactive resource allocation method based on the prediction of resource requests. Figure 1 shows the implement process of this method. First, a RT-based adaptive prediction method is used to forecast the future resource requests based on the past data of

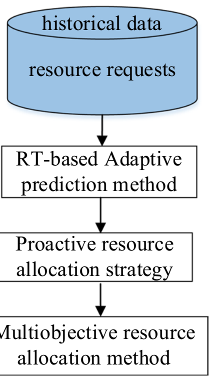
historical data
resource requests
RT-based Adaptive prediction method
Proactive resource allocation strategy
Multiobjective resource allocation method

Fig. 1 Implementation process of a proactive resource allocation. First, a RT-based adaptive prediction method is used to forecast the future resource requests based on the past data of resource requests. Then, a proactive resource allocation strategy is proposed based on the prediction of resource requests. Finally, a multiobjective resource allocation method is proposed and solved by an improved NSGA-II algorithm

resource requests. Then, a proactive resource allocation strategy is proposed based on the prediction of resource requests. Finally, a multiobjective resource allocation method is proposed and solved by an improved NSGA-II algorithm.

3.1 RT-based adaptive prediction method

The prediction method is designed with two goals: reduce the prediction time and improve the prediction accuracy. The prediction procedure is shown in Fig. 2. The m component sequences are extracted from a VM request sequence using the principal component analysis method. Next, these component sequences are detected to find and preprocess the outliers. Then, the RT values of these preprocessed sequences are calculated. Finally, these sequences are predicted by adaptively selecting the EEMD-ARIMA or EEMD-RT-ARIMA method according to the comparison between their RT values and the thresholds.

A cloud platform provides many VM flavors, such as 2CPU4G (2 CPU cores, 4G memory) and 4CPU8G (4 CPU cores, 8G memory). We cannot predict each type of VM requests due to the high prediction time. Therefore, principal component analysis is first used in our prediction method, which can extract the major component sequences to reduce the prediction time. For example, a VM request sequence with n types of VMs is denoted as $S =< s_1, ..., s_i, ..., s_k >$, where s_i represents the VM number of the ith request. A component sequence S_l = $<$ s_{l1},...,s_{li},...,$s_{lk}$$>$ can be extracted from this sequence S for the VM type l, where s_{li} denotes the VM number of the ith request. Thus, an original VM sequence can be divided into many component sequences. We can select the fewest component sequences to implement the prediction of VM requests, where the ratio of the sum of their VM requests to the total number of VM requests (it is called the proportion of VM requests) is beyond the predefined threshold T_{th} at each sampling point. These components sequences can be regarded as the major component sequences. For example, there are two component sequences $S_h =< s_{h1}, ..., s_{hi}, ..., s_{hk} >$ and $S_g =< s_{g1}, ..., s_{gi}, ..., s_{gk} >$, where s_{hi} and s_{gi} are the quantities of the different types of VM requests. For $\forall s_{hi} \in S_h$ and $s_{gi} \in S_g$,

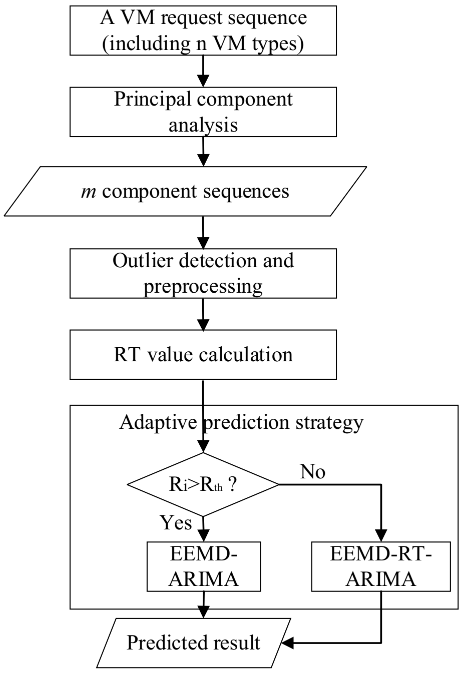
A VM request sequence (including n VM types)
Principal component analysis
m component sequences
Outlier detection and preprocessing
RT value calculation
Adaptive prediction strategy
Ri>Rth ?
No
Yes
EEMD-ARIMA
EEMD-RT-ARIMA
Predicted result

Fig. 2 RT-based adaptive prediction method. The m component sequences are extracted from a VM request sequence using the principal component analysis method. Next, these component sequences are detected to find and preprocess the outliers. Then, the RT values of these preprocessed sequences are calculated. Finally, these sequences are predicted by adaptively selecting the EEMD-ARIMA or EEMD-RT-ARIMA method according to the comparison between their RT values and the thresholds

$$\text{if } (s_{hi} + s_{gi})/s_i \geq T_{th}, \quad select(S_h, S_g) \rightarrow S_{main} \tag{1}$$

These two component sequences are selected into a set S_{main} with the major component sequences to implement the prediction. Intuitively, the higher the threshold T_{th}, the more the selected component sequences. The prediction of VM requests is more accurate, which ensures resource allocation to be more correct. However, the more the component sequences, the higher the prediction cost. For instance, if the threshold is set as T'_{th} higher than the one T_{th}, that is $T'_{th} > T_{th}$, three major component sequences S_h,S_g and S_l may be selected into the set S_{main}. For $\forall s_{hi} \in S_h$, $s_{gi} \in S_g$ and $s_{li} \in S_l$.

$$\text{if } (s_{hi} + s_{gi} + s_{li})/s_i \geq T'_{th}, \quad select(S_h, S_g, S_l) \rightarrow S_{main} \tag{2}$$

.

Then, each component sequence is decomposed into many subsequences to perform the prediction using EEMD-ARIMA or EEMD-RT-ARIMA method. Supposing a component sequence is decomposed into m subsequences and each subsequence cost n seconds to finish the prediction. Here, the running time of each prediction is almost identical, we suppose that it is n seconds for each subsequence. If two component sequences S_h and S_g are selected under the threshold T_{th}, the prediction cost can be calculated as follows.

$$C\left(S_h, S_g, T_{th}\right) = 2m \cdot n \tag{3}$$

However, if three component sequences S_h, S_g and S_l are selected under the threshold T'_{th}, the prediction cost will be calculated as follows.

$$C'\left(S_h, S_g, S_l, T'_{th}\right) = 3m \cdot n \tag{4}$$

It can be seen that the prediction cost will greatly increase though more component sequences selected by setting a higher threshold can improve the prediction accuracy. This will cause a delayed resource allocation not to ensure the normal running of applications. Therefore, setting the threshold T_{th} is important, which not only need to reflect the major VM requests but also reduce the prediction time cost. Supposing p major component sequences have been selected to predict the future VM requests and a unselected component sequence S_l has more VM requests than other component sequences. The threshold T_{th} can be set as an approximation of the minimum value of the proportion of VM requests according to the following formula when one of two conditions is satisfied. The threshold T_{th} impact the prediction accuracy and the prediction time cost.

$$T_{th} \leftarrow \min\left\{(s_{11} + \ldots + s_{1k})/s_1, \ldots, (s_{i1} + \ldots + s_{ik})/s_i \ldots, \left(s_{p1} + \ldots + s_{pk}\right)/s_p\right\} \tag{5}$$

$$\text{S.T.} \quad s_{li}/s_i < \varepsilon_l \tag{6}$$

$$\text{or} \quad 1/(p+1) > \varepsilon_t \tag{7}$$

ε_l and ε_t indicate a threshold of the proportion of VM requests and a threshold of the ratio of prediction time cost, respectively. When both the added proportion of VM requests s_{li}/s_i is beyond this threshold ε_l and the added ratio of prediction time cost $1/(p+1)$ is below this threshold ε_t, the component sequence S_l will be selected to predict the future VM requests.

The quartile method is adopted to detect the outlier points of these major component sequences. Firstly, we calculate the first quartile $Q1$, the third quartile $Q3$ and the interquartile range (IQR) by the formula $IQR = Q3 - Q1$, detect the outliers more than 1.5 times over $Q3$ or less than 1.5 times below $Q1$ and finally replace these outliers via a cubic spline interpolation method.

The preprocessed component sequences are executed using RT method. Then, we set up an adaptive prediction method based on the RT (APMRT). If the RT value of a component sequence is higher than a predefined threshold R_{th}, the EEMD-ARIMA method is selected to predict the future resource requests. Otherwise, the EEMD-RT-ARIMA is selected to make the prediction. Thus, the prediction accuracy can be improved by preprocessing the outliers of the major component sequences and selecting a more accurate prediction method. We can determine the future number of each type of VM requests and proactively allocate resources to guarantee the timeliness of the resource allocation.

In this method, the time complexity of extracting a component sequence is $O(k)$. Thus, the time complexity of extracting m component sequences and data preprocessing becomes $O(m \cdot k)$. Then, the RT values of the extracted sequences are calculated and predicted by using the adaptive prediction algorithm APMRT. The time complexity becomes $O(2m \cdot k + q \cdot m \cdot k)$ or $O(2m \cdot k + p \cdot m \cdot k)$, where q,p are separately the number of the decomposed component sequences and the new component sequences reconstructed ($p < q$). Therefore, the time complexity of the APMRT algorithm is $O(Q \cdot m \cdot k)$ or $O(P \cdot m \cdot k)$, which is less than the time complexity $O(Q \cdot n \cdot k)$ or $O(P \cdot n \cdot k)$ of all n component sequences. The prediction time is largely reduced by extracting the main component sequences.

3.2 Proactive resource allocation strategy

A cloud resource allocation algorithm should actively predict the future resource requests and allocate resources in advance to cope with the sudden increase of resource requests in the future. The proactive resource allocation framework is shown in Fig. 3. The RT-based adaptive prediction method is used to predict the future number of VM requests based on past data. A hybrid VM request queue is formed by combining the future VM requests predicted with the current VM requests.

Suppose that the current VM request sequence is denoted as $V(t) =< v_1(t), ..., v_i(t), ...v_n(t) >$, where $v_i(t)$ indicates the VM number of the i th request at time t. The number $D(t+h)$ of the future l major types of VM requests at time $t+h$ predicted via the adaptive prediction method APMRT is denoted as follows.

$$D(t+h) = D^1(t+h) + ... + D^i(t+h) + ... + D^l(t+h) \tag{8}$$

$D^i(t+h)$ is the i th major type of VM requests at time $t+h$. The total number of VM requests $N(t)$ at t time should be the sum of the current number of VM requests $V(t)$ and the predicted number of VM requests $D(t+h)$ as follows.

$$N(t) = M(t) + D(t+h) \cdot C(t) \cdot P(t) \tag{9}$$

$M(t)$=$v_1(t) + ... + v_i(t) + ... + v_n(t)$ is the current number of VM requests at t time. If the predicted number of VM requests $D(t+h)$ is not less than the threshold N_{th}, some VMs should be allocated resources in advance. The parameter $C(t)$ should equal to 1 and $P(t)$ is a percentage (e.g., 30%) of VM requests to be allocated resources in advance with respect to the predicted number of VM requests $D(t+h)$. Otherwise, it does not need to provide VMs in advance, that is, $C(t) = 0$.

After the predicted number of VM requests $D(t+h)$ is determined, the VM request sequence should be established. Assuming that the predicted number of VM requests is

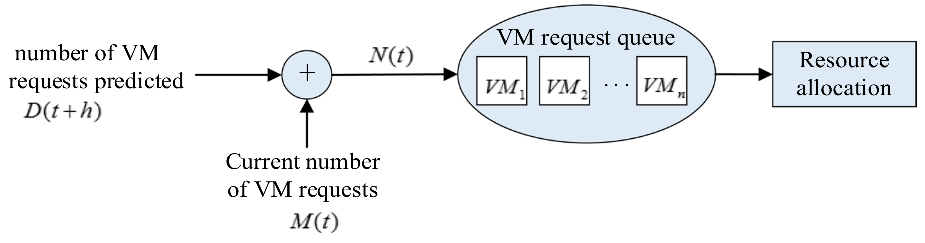
number of VM requests predicted $D(t+h)$
+
$N(t)$
VM request queue
VM_1 VM_2 $\cdots$ VM_n
Resource allocation
Current number of VM requests $M(t)$

Fig. 3 Proactive resource allocation strategy. The RT-based adaptive prediction method is used to predict the future VM requests based on past data. A hybrid VM request queue is formed by combining the future VM requests predicted actively with the current VM requests

ordered in descending order from VM type 1 to l, the largest VM requests (i.e., the type 1 VM requests) are placed at the front of the VM request sequence, and the smallest VM requests (i.e., the type l VM requests) are placed at the end of the VM request sequence. The predicted VM request sequence can be expressed as follows.

$$V(t+h) =< v_1^1(t+h), v_2^1(t+h), v_3^1(t+h), ..., v_j^i(t+h), v_{j+1}^i(t+h)...v_m^l(t+h) > \tag{10}$$

$v_j^i(t+h)$ and $v_{j+1}^i(t+h)$ are the quantities of the j th and j+1 th VM requests with the same VM type i. Thus, the VM request sequence at time t can be expressed as follows.

$$V'(t) =< v_1(t), ...v_n(t), v_1^1(t+h), v_2^1(t+h), v_3^1(t+h), ..., v_j^i(t+h), v_{j+1}^i(t+h), ..., v_m^l(t+h) > \tag{11}$$

3.3 Multiobjective resource allocation method

Our previous work has presented a multiobjective resource allocation method [45]. This method builds a multiobjective function with the minimum number of the used PMs $\min\{\sum_S x_{ij}\}$ and the minimum total resource performance matching distance between VMs and PMs $\min\{\sum_S MD_{ij}\}$, where x_{ij} denotes the mapping element between the VM v_i and the PM p_j. If the VM v_i is placed on the PM p_j, x_{ij} equals 1. Otherwise, x_{ij} equals 0. Thus, the formula $\sum_S x_{ij}$ represents the total number of the used PMs under a solution S. In the formula of the resource performance matching distance $MD_{ij} = \sqrt{\sum_{k=1}^{3} (npv_{ik} - npp_{jk})^2}$, npv_{ik} represents the normalized resource performance variable of VM v_i, npp_{jk} represents the corresponding normalized resource performance variable of the PM p_j and k=1, 2, 3 denote the CPU, memory and disk resources, respectively.

This paper proposes a new resource allocation method based on the prediction of VM requests (RAMPVR), which further considers two issues to improve the previous resource allocation method. One is to reduce the waste of physical resources. If the proportion of different types of resources from a VM request is closer to those free resources of a PM, it is less likely to cause resource waste for this PM. That is, the closer the resource proportion $v_{i1} : v_{i2} : v_{i3}$ of a VM is to that $p_{j1} : p_{j2} : p_{j3}$ of a PM, the lower the resource waste, where v_{i1}, v_{i2} and v_{i3} represent the requested number of CPU cores, memory capacity and disk size of the VM v_i, respectively; and p_{j1}, p_{j2} and p_{j3} denote the free number of CPU cores, memory capacity and disk size of the PM p_j, respectively. Therefore, we build the resource proportion matching distance model shown in formula (12), where p_{jk} and v_{ik} represent the free capacity of resource type k of the PM p_j and the requested resource capacity of the VM v_i, respectively, and R_k denotes the coefficient that adjusts the imbalanced values of parameter $H = p_{jk} \cdot v_{i1}/p_{j1} - v_{ik}$ for different resource types. For instance, if the values of the parameter H for CPU and disk resources are 2 and 200, the disk will become the dominant resource. Therefore, the adjustment coefficient R_k for the disk resource should be adjusted to a lower value than that for the CPU resource, such as using $R_k = 1$ for the CPU resource and $R_k = 0.1$ for the disk resource.

$$MPM_{ij} = \sqrt{\sum_{k=1}^{3}\left(\left(\frac{p_{jk}\cdot v_{i1}}{p_{j1}} - v_{ik}\right)\cdot R_k\right)^2} \tag{12}$$

Thus, we set up a multiobjective optimization problem of resource allocation according to the number of the used PMs $\sum_S x_{ij}$, the total resource performance matching distance $\sum_S MD_{ij}$ and the total resource proportion matching distance $\sum_S MPM_{ij}$ as follows.

$$M : \min\left\{\sum_S x_{ij}\right\} \tag{13}$$

$$\min\left\{\sum_S MD_{ij}\right\} \tag{14}$$

$$\min\left\{\sum_S MPM_{ij}\right\} \tag{15}$$

The first goal of the multiobjective optimization problem M of resource allocation is to minimize the total number of the used PMs, as shown in formula (13), which depends on the value of each mapping element x_{ij} between the VM v_i and the PM p_j under a solution S. The second goal of the problem M is to minimize the total resource performance matching distance under a solution S, as shown in formula (14), which depends on the resource performance matching distance MD_{ij} between the VM v_i and the PM p_j. The third goal of the problem M is to minimize the total resource proportion matching distance under a solution S, as shown in formula (15), which depends on the resource proportion matching distance MPM_{ij} between the VM v_i and the PM p_j. In addition, the total CPU, memory and disk capacities requested by the VMs placed on PM p_j are less than its free CPU, memory and disk capacities, respectively. Thus, the constraint conditions are shown in formulas (16), (17) and (18), respectively.

$$\text{S.T.} \quad \sum_S v_{i1}\cdot x_{ij} \le p_{j1} \tag{16}$$

$$\sum_S v_{i2}\cdot x_{ij} \le p_{j2} \tag{17}$$

$$\sum_S v_{i3}\cdot x_{ij} \le p_{j3} \tag{18}$$

Another is to optimize the solution algorithm that accelerates the solution speed of the multiobjective optimization function. The NSGA-II is a classical algorithm for solving a multiobjective optimization problem [46–48]. As a Nondominated Sorting Genetic Algorithm, it has been widely applied in solving the multiobjective problem and achieves good effectiveness [39–41]. However, the NSGA-II algorithm has a problem that the computation time of the fitness values (i.e., the objective functions) is too long to ensure the timelessness of the resource allocation. Furthermore, the fitness values of a large number of individuals need to be calculated in the population evolution. Hence, we will improve the NSGA-II algorithm to accelerate the solution speed using the parallel computation of the fitness function. We adopt multicore processors to calculate the fitness values of the individuals in parallel, which can accelerate the convergence of the proposed algorithm. The fitness values of each individual are calculated as follows.

$$f_1(I_k) = \sum_S x_{ij} \tag{19}$$

$$f_2(I_k) = \sum_S MD_{ij} \tag{20}$$

$$f_3(I_k) = \sum_S MPM_{ij} \tag{21}$$

4 Experiments and analysis

4.1 Prediction of VM requests

We select two time series $S1$ and $L1$ of continuous container requests, which are taken from the Alibaba cluster data [49] as the experimental dataset of VM requests. These time series only include the data on CPU and memory resources. We will use the sequence $S1$ as an example to illustrate the adaptive prediction process. This sequence $S1$ includes 95 sampling points (475 min) and 28 types of VMs, where each sampling point counts the total number of VMs in a 5-min period. We use the principal component analysis method to extract its component sequences $S2$ and $S3$ and calculate the threshold T_{th}=85% according to the predefined thresholds ε_l=5% and ε_t=20% and formula (5)–(7). These sequences are all shown in Fig. 4, where the sequences $S2$ and $S3$ represent the VM numbers for the types of 4-core CPU and 1.56 memory (CPU = 400 means 4-core CPU and mem = 1.56 means 1.56 memory) and 8-core CPU and 3.13 memory (CPU = 800 and mem = 3.13), respectively. It is noted that the number of CPU cores and amounts of memory are normalized.

It can be observed that the number of VM requests dynamically changes and demonstrates the characteristic of suddenness, which makes the future resource requests difficult to predict. It can also be seen that the sequence $S2$ with the 4-core CPU and 1.56 memory is consistent with the trend of the sequence $S1$. The sequence $S3$ with the 8-core CPU and 3.13 memory is roughly the same as the trend of the sequence $S1$, but they have some differences in the detailed fluctuations.

Next, we use the quartile method to detect the outliers with red "+" shown in Fig. 5. Sequentially, they are replaced by new data generated by a cubic spline interpolation method. Thus, we get the preprocessed sequences shown in Fig. 6.

Then, we use the adaptive prediction method APMRT to implement the prediction for these preprocessed sequences. The RT values of these sequences are first calculated. And the threshold R_{th} is set as 20, which can be roughly observed from the experimental testing for Alibaba cluster data [6]. It is noted that this threshold R_{th} is different for different traces or scenarios, which need to be achieved from the experimental testing or

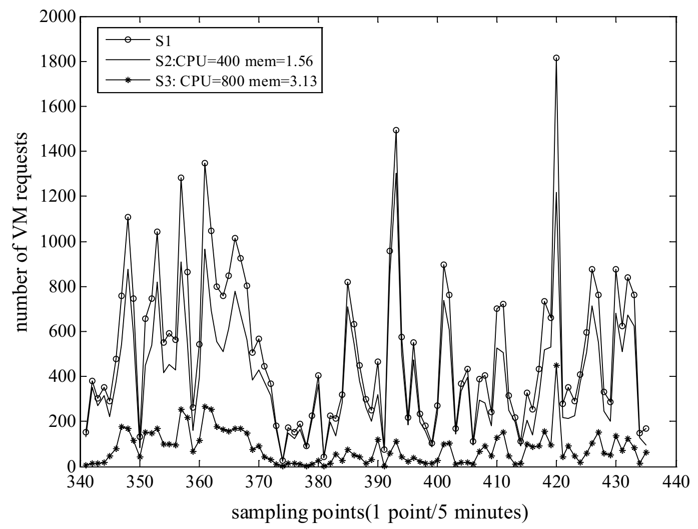

Fig. 4 Number of new VMs created for online services. The S1 curve depicts the total number of 28 types of new VMs. The S2 and S3 curves depict the quantities of new VMs for the type of CPU = 400 and mem = 1.56 and the type of CPU = 800 and mem = 3.13, respectively

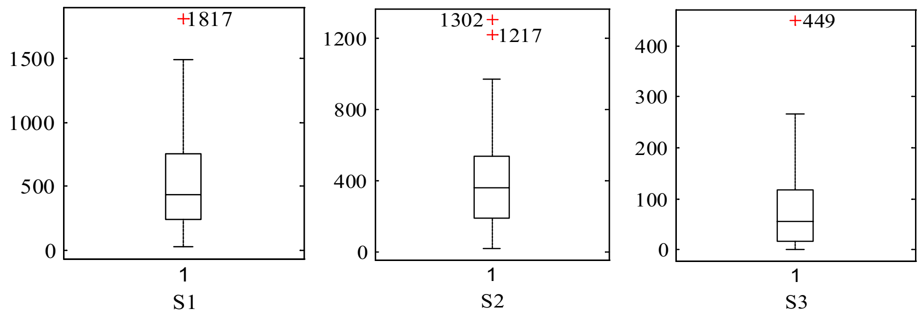

Fig. 5 Box-plot of VM request sequences. The first, second and third subgraphs demonstrate the outliers with red "+" of the S1, S2 and S3 sequences, respectively

via your expert experience. We select the first 80 sampling points as the training data and the next 5 points, 10 points, and 15 points as the testing data, respectively. When the RT value of a sequence is lower than the predefined threshold R_{th}, the EEMD-RT-ARIMA method is selected to execute the prediction; otherwise, the EEMD-ARIMA method is selected.

Figure 7 shows the mean absolute percentage error (MAPE) of the prediction results. It can be seen that the MAPEs of the 10-point and 15-point predictions increase greatly compared with those of the 5-point prediction. For instance, the EEMD-RT-ARIMA method achieves a MAPE of 9.87% for the 5-point prediction of the sequence $S1$, but it achieves MAPEs of 29.62% and 54.99% for the 10-point prediction and 15-point prediction, respectively. Similarly, the EEMD-ARIMA method achieves a MAPE of 11.28% for the 5-point prediction of the sequence $S2$, but its MAPEs are 38.31% and 64.51% for the 10-point prediction and 15-point prediction, respectively. It implies that both methods are not suitable for long-term prediction but are for short-term prediction. The

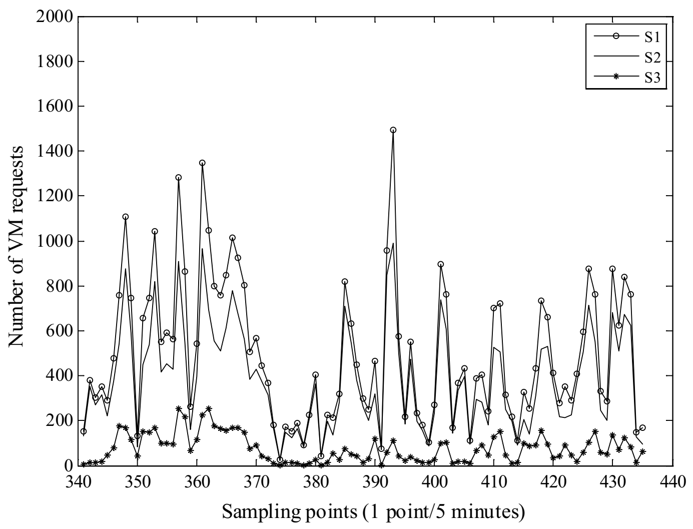

Fig. 6 The preprocessed sequences of VM requests. The S1, S2 and S3 curves represent the preprocessed sequences after using a cubic spline interpolation method to replace the outliers of the original S1, S2 and S3 sequences, respectively

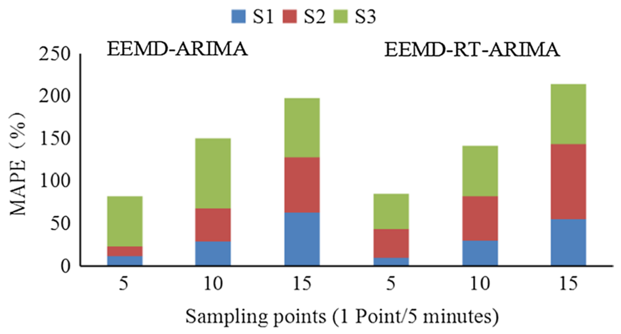

Fig. 7 The MAPEs of adaptive prediction for different types of VM request sequences. The first three columns represent the MAPEs of the 5-point, 10-point and 15-point prediction obtained by EEMD-ARIMA method. The last three columns represent those obtained by EEMD-RT-ARIMA method. The blue, red and green parts represent the MAPEs of the S1, S2 and S3 sequences in each column, respectively

reason is mainly due to the strong fluctuation of the sampling data in a short time. The EEMD-RT-ARIMA method achieves lower MAPEs than the EEMD-ARIMA method for the sequences $S1$ and $S3$, while it is the opposite for the sequence $S2$. We find that the RT values of S1-S3 are 19, 21 and 19, respectively. This indicates that the proposed prediction method is effective. When a sequence has strong fluctuation, the cumulative prediction error of the component sequences obtained by EEMD decomposition can be less than that caused by the non-stationary sequence. Thus, EEMD-ARIMA method can achieve higher prediction accuracy. Otherwise, EEMD-RT-ARIMA method can reduce more prediction error accumulation than EEMD-ARIMA, and thus, it can achieve higher prediction accuracy. Figure 8 depicts the future 5-point values predicted via the proposed APMRT method. In the same way, we predict the future 5-point values for the sequence $L1$, as in Fig. 9. It is also possible that other factors impact the prediction

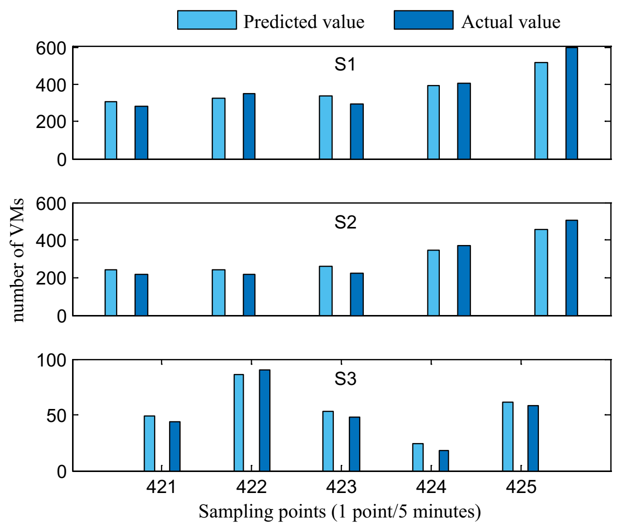

Fig. 8 Number of the predicted VMs for the S1, S2 and S3 sequences. The first, second and third subgraphs depict the comparison between the actual values and the predicted 5-point values obtained by our proposed method APMRT for the preprocessed S1, S2 and S3 sequences, respectively

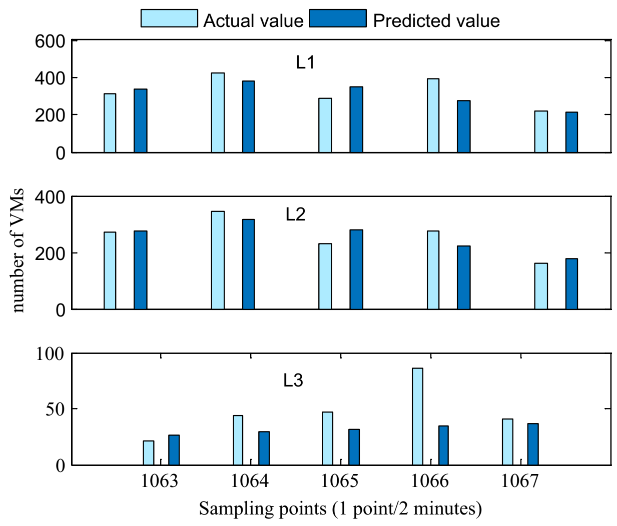

Fig. 9 Number of the predicted VMs for the L1, L2 and L3 sequences. The first, second and third subgraphs depict the comparison between the actual values and the predicted 5-point values obtained by our proposed method APMRT for the preprocessed L1, L2 and L3 sequences, respectively

accuracy. In the future, we will further study this issue. This paper pays more attention on virtual resource allocation based on an adaptive prediction of resource requests.

4.2 Simulation of the resource allocation

As shown in Fig. 8, 519 VMs (4-core CPU and 1.56 memory) and 62 VMs (8-core CPU and 3.13 memory) are predicted for 425 sampling points. There is sudden growth over 300. Therefore, we should allocate resources for some VMs in advance to alleviate the latency of the resource allocation at 424 sampling points. We set up the ratio of the proactive resource allocation $P_i = 0.3$. Thus, the number of VMs that needs to be created at 424 sampling points can be calculated by the formula 408 + (457 + 62)*0.3 = 564. The number of available PMs is 2972. The resource allocation problem becomes the problem of creating 564 VMs on 2972 available PMs. Similarly, we can observe that the method predicts 281 VMs (4-core CPU and 1.56 memory) and 31 VMs (8-core CPU and 3.l3 memory) for 1065 sampling points for the sequence $L1$ from Fig. 9. If we set the ratio of the proactive resource allocation $P_i = 1/3$, we should create $423 + (281 + 31)/3 = 527$ VMs on 2972 PMs at 1064 sampling points.

Even if the prediction may fail, the proactive resource allocation will not be greatly affected. For example, if the predicted number of VM requests is 893 or 297 for 425 sampling point, the MAPE will exceed 50%, that is, the prediction fails. We should create 268 or 89 VMs more than the original number of 408 according to the proactive resource strategy for 424 sampling points. It is not more than the actual number of VM requests for 425 sampling points. However, the more the proactive number of VM requests is, the longer the resource allocation time is for 424 sampling points. Therefore, the prediction error should be limited in a certain range.

To verify the effectiveness of the proposed RAMPVR method, we adopt these following metrics to compare our method with others.

(1) Number of the used PMs: If the number of the used PMs is less, some idle PMs can be closed to reduce the energy consumption and cost.
(2) Resource performance matching distance: The smaller the resource performance matching distance is, the better the VMs match with the PMs regarding their resource performance.
(3) Resource proportion matching distance: The smaller the resource proportion matching distance is, the less the resource waste.
(4) Resource utilization: A good resource allocation method should maximize and homogenize each type of resource utilization.
(5) Time cost of resource allocation: Our prediction-based resource allocation method reduces the VM creation time by allocating resources for the future VM requests in advance. This paper mainly focused on reducing the solving time of this method. The lower the solving time is, the more resource allocation time that is reduced.

We set the population size, the crossover probability, the crossover distribution index and the mutation distribution index as 200, 0.85, 20 and 20, respectively; and set the reciprocal of the number of variables as the mutation probability in the simulation. The maximum evaluation times of the fitness values and the maximum number of iterations of populations are set as 20,000 and 100, respectively. We compare the proposed RAMPVR method with the round robin (RR), SPEA2 and NSGA-II methods in terms of the number of the used PMs, the resource performance matching distance, the resource proportion matching distance, the resource utilization and the solving time. SPEA2 is another presentative elite multi-objective evolutionary algorithm [50], which can obtain multiple pareto optimal solutions in a single run. It has been widely used in different domain [41, 52] and has become the standard for performance comparison of multi-objective evolutionary algorithms [53, 54]. Each

Table 2 Experimental results of resource allocation for S1 sequence

Metrics	RR	SPEA2	NSGA-II	RAMPVR	
				8 Threads	10 Threads
Number of used PMs	564	477	473	460	462
Resource performance matching distance	352.20	435.72	402.57	407.57	395.27
Resource proportion matching distance	18,769.42	11,299.49	12,319.08	11,975.88	12,521.97
Resource utilization (CPU)	74.90%	58.62%	64.01%	62.80%	64.79%
Resource utilization (memory)	28.76%	60.28%	65.45%	64.29%	66.20%
Solution time	0.3 s	1593 s	1551 s	977 s	886 s

Table 3 Experimental results of resource allocation for L1 sequence

Metrics	RR	SPEA2	NSGA-II	RAMPVR	
				8 Threads	10 Threads
Number of used PMs	527	423	435	430	426
Resource performance matching distance	338.46	290.66	295.06	292.39	303.66
Resource proportion matching distance	17,464.66	11,071.06	11,541.04	11,508.03	10,608.15
Resource utilization(CPU)	73.12%	63.58%	63.13%	64.15%	61.53%
Resource utilization(memory)	27.68%	65.04%	64.61%	65.58%	63.07%
Solution time	0.3 s	1592 s	1375 s	821 s	777 s

method is executed 10 times and the respective average results are computed. The experimental results are shown in Tables 2 and 3.

It can be seen from Table 2 that the SEPA2, NSGA-II and RAMPVR methods use different numbers of PMs. Even the RAMPVR method uses different numbers of PMs in different experiments, such as 460 and 462 PMs. The less the number of the used PMs is, the more the saved resource cost is. The CPU and memory utilization of the used PMs are more balanced via resource proportion matching, which will reduce the resource waste. The SPEA2 method achieves CPU utilization of 58.62% and memory utilization of 60.28%, the NSGA-II method obtains 64.01% and 65.45%, and the RAMPVR method achieves 62.80% and 64.29% under the parallel computing of 8 threads, respectively. In addition, they basically keep a similar number of the used PMs, similar resource performance matching and similar resource proportion matching because they achieve the trade-off among them. The RR method demonstrates big differences in these aspects. It uses the most PMs because it adopts a polling mechanism. Furthermore, it achieves the highest resource performance, the highest proportion matching distances, and the most unbalanced resource utilization with CPU utilization of 74.90% and memory utilization of 28.76% due to the polling mechanism, which will cause high resource waste. However, it has a lower solution time of only 0.3 s because it uses a simple heuristic algorithm to solve the problem. Compared with the SPEA2 and NSGA-II methods, the RAMPVR method uses less time to solve the multiobjective functions. For instance, the SPEA2 and NSGA-II methods, respectively, use 1593 and 1551 s to solve the multiobjective problem, but the RAMPVR method only costs 886 s to solve it with the parallel computing of 10 threads. Table 3 also demonstrates this situation. Thus, the VM creation time can be greatly reduced according to the time saved by predicting the VMs in advance. The timelessness and rapidness of resource allocation can be guaranteed.

5 Conclusion

Cloud resource requests demonstrate the characteristics of being diverse, arriving in bursts and being uncertain, which causes the resource allocation to lag behind the resource requests and the quality of service not to be ensured in a cloud platform. This paper proposes a multiobjective resource allocation method based on an adaptive prediction method for resource requests. This method can allocate virtual resources in advance to alleviate the delay problem of resource provision by using an adaptive method to predict the future resource requests. The timelessness of the resource allocation is further guaranteed by improving the NSGA-II algorithm to reduce the solving time of the multiobjective optimization problem. In addition, the various types of resources in a PM are evenly utilized, which reduces resource waste. Two experiments are conducted to verify the effectiveness of our proposed method. The experimental results show that this method realizes the balance between CPU and memory resources and reduces the resource allocation time by at least 43% (10 threads) compared with the SPEA2 and NSGA-II methods.

Abbreviations

QoS: Quality of service; VM: Virtual machine; GOA: Grasshopper optimization algorithm; PCGWO: Performance-cost grey wolf optimization; SWA: Single weight algorithm; DWA: Double weight algorithm; NSGA-II: Nondominated Sorting Genetic Algorithm with the Elite Strategy; SLA: Service level agreement; GA: Generic algorithm; ARIMA: Autoregressive Integrated Moving Average Model; SES: Simple exponential smoothing; LR: Linear regression; PM: Physical machine; BPNN: Back Propagation Neural Network; SVR: Support vector regression; EEMD: Ensemble empirical mode decomposition; RT: Runs test; IQR: Interquartile range; APMRT: An adaptive prediction method based on RT; RAMPVR: Resource allocation method based on the prediction of VM requests; MAPE: Mean absolute percentage error.

Acknowledgements
Not applicable.

Authors' contributions

Jing Chen is a major contributor in proposing the method, implementing simulation and writing a manuscript. Yinglong Wang gave some ideas and suggestions and revised this manuscript. Tao Liu carried out the partial experimental work and data analysis. All authors read and approved the final manuscript.

Funding
This work was supported in part by National Key Research and Development Project of China, Project No. 2018YFB1404501, in part by Shandong Provincial Natural Science Foundation under Grant ZR2020MF034, ZR2019QF014 and ZR2019PF015, in part by the Youth Science Funds of Shandong Academy of Sciences under Grant 2019QN0025, in part by the "Colleges and Universities 20 Terms" Foundation of Jinan City, China, under Grant 2018GXRC015.

Availability of data and materials

The datasets used during the current study are available from the corresponding author on reasonable request.

Competing interests

The authors declare that they have no competing interests.

Received: 7 February 2020 Accepted: 28 January 2021
Published online: 06 February 2021

References

1. P. Pradhan, P.K. Behera, N.N.B. Ray, Modified round robin algorithm for resource allocation in cloud computing. Proc. Comput. Sci. **85**, 878–890 (2016)
2. S. Shirvastava, R. Dubey, M. Shrivastava, Best fit based VM allocation for cloud resource allocation. Int. J. Comput. Appl. **158**(9), 25–27 (2017)
3. M. Katyal, A. Mishra, Application of selective algorithm for effective resource provisioning in cloud computing environment. Int. J. Cloud Comput. Serv. Archit., 4(1), 1–10(2014).

4. X. Chen, J.X. Lin, Y. Ma et al., Self-adaptive resource allocation for cloud-based software services based on progressive QoS prediction model. Sci. China Inf. Sci. **62**(11), 1–3 (2019)
5. J. Chen, Y. Wang, A resource request prediction method based on EEMD in cloud computing. Proc. Comput. Sci. **131**, 116–123 (2018)
6. J. Chen, Y. Wang, A hybrid method for short-term host utilization prediction in cloud computing. J. Electr. Comput. Eng. **2782349**, 1–14 (2019)
7. D. Shen, Research on application-aware resource management for heterogeneous big data workloads in cloud environment. Dongnan University, 2018.
8. X. Chen, J. X. 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. Exp., **31**(23), e4463 (2019).
9. K. Gurleen, B. Anju, A survey of prediction-based resource scheduling techniques for physics-based scientific applications, Mod. Phys. Lett. B, 32(25), 1850295(2018).
10. Y.J. Laili, S.S. Lin, D.Y. Tang, Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment. Robot. Comput. Integr. Manuf. **61**, 101850 (2020)
11. K. Reihaneh, S.E. Faramarz, N. Naser, M. Mehran, ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments. J. Supercomput. **73**(6), 2430–2455 (2017)
12. K. Kavitha, S. C. Sharma, Performance analysis of ACO-based improved virtual machine allocation in cloud for IoT-enabled healthcare. Concurr. Comput. Pract. Exp., e5613 (2019).
13. J. Vahidi, M. Rahmati, in IEEE 5th Conference on Knowledge Based Engineering and Innovation (KBEI). Optimization of resource allocation in cloud computing by grasshopper optimization algorithm, pp. 839–844 (2019).
14. U. Rugwiro, C.H. Gu, W.C. Ding, Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning. J. Internet Technol. **20**(5), 1463–1475 (2019)
15. S. Shenoy, D. Gorinevsky, N. Laptev, Probabilistic Modelling of Computing Request for Service Level Agreement. IEEE Trans. Serv. Comput. **12**(6), 987–993 (2019)
16. Z.H. Liu, Z.J. Wang, C. Yang, Multi-objective resource optimization scheduling based on iterative double auction in cloud manufacturing. Adv. Manuf. **7**(4), 374–388 (2019)
17. A. A. Motlagh, A. Movaghar, A. M. Rahmani, Task scheduling mechanism in cloud computing: a systematic review. Int. J. Commun. Syst. e4302 (2019).
18. M. Kumar, S.C. Sharma, A. Goel, S.P. Singh, A comprehensive survey for scheduling techniques in cloud computing. J. Netw. Comput. Appl. **143**, 1–33 (2019)
19. N. D. Vahed, M. Ghobaei-Arani, A. Souri, Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review. Int. J. Commun. Syst. **32**(14), e4068 (2019).
20. F. Sheikholeslami, N. J. Navimipour, Auction-based resource allocation mechanisms in the cloud environments: a review of the literature and reflection on future challenges. Concurr. Computat. Pract. Exp., **30**(16), e4456 (2018).
21. G. Natesan, A. Chokkalingam, An improved grey wolf optimization algorithm based task scheduling in cloud computing environment. Int. Arab J. Inf. Technol. **17**(1), 73–81 (2020)
22. M. A. Reddy, K. Ravindranath, Virtual machine placement using JAYA optimization algorithm. Appl. Artif. Intell. https://doi.org/10.1080/08839514.2019.1689714.
23. S. Souravlas, S. Katsavounis, Scheduling fair resource allocation policies for cloud computing through flow control. Electronics **8**(11), 1348 (2019).
24. L. Guo, P. Du, A. Razaque, et al. IEEE 2018 Fifth international conference on software defined systems (SDS). Energy saving and maximize utilization cloud resources allocation via online multi-dimensional vector bin packing (2018), pp. 160–165.
25. N. Gul, I. A. Khan, S. Mustafa, o. Khalid, A. U. R. Khan, CPU-RAM-based energy-efficient resource allocation in clouds. J. Supercomput. **75**(11), 7606–7624 (2019).
26. R.L. Sri, N. Balaji, An empirical model of adaptive cloud resource provisioning with speculation. Soft. Comput. **23**(21), 10983–10999 (2019)
27. J. J. Prevost, K. M. Nagothu, B. Kelley, et al., in 6th International Conference on System of Systems Engineering (SoSE). Prediction of cloud data center networks loads using stochastic and neural models. (2011), pp. 276–281.
28. H.L. Tang, C.L. Li, J.P. Bai, J.H. Tang, Y.L. Luo, Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud-edge environment. Comput. Commun. **134**, 70–82 (2018)
29. Y. Wang, Y. Guo, Z. Guo, T. Baker, W. Liu, CLOSURE: A cloud scientific workflow scheduling algorithm based on attack-defense game model. Future Gener. Comput. Syst. **111**, 460–474 (2020)
30. M. Al-khafajiy, T. Baker, M. Asim et al., COMITMENT: a fog computing trust management approach. J. Parallel Distrib. Comput. **137**, 1–16 (2020)
31. T. Bakera, E. Ugljaninb, N. Facic et al., Everything as a resource: Foundations and illustration through Internet-of-things. Comput. Ind. **94**, 62–74 (2018)
32. G. Ismayilov, H.R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Future Gener. Comput. Syst. **102**, 307–322 (2020)
33. K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
34. S. Jeddi, S. Sharifian, A water cycle optimized wavelet neural network algorithm for request prediction in cloud computing. Cluster Comput. **22**(4), 1397–1412 (2019)
35. F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C.M. Leung, Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. IEEE Syst. J. **12**(2), 1688–1699 (2018)
36. R. Shaw, E. Howley, E. Barrett, An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. Simul. Model. Pract. Theory **93**, 322–342 (2019)
37. H. Mehdi, Z. Pooranian, P. G. V. Naranjo. Cloud traffic prediction based on fuzzy ARIMA model with low dependence on historical data. Trans. Emerg. Telecommun. Technol. e3731 (2018).

38. Zharikov, S. Telenyk, P. Bidyuk, Adaptive workload forecasting in cloud data centers. J. Grid Comput. https://doi.org/10.1007/s10723-019-09501-2.
39. M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, E. Agiatzidou, Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. Future Gener. Comput. Syst. **93**, 442–459 (2019)
40. C. Li, H. Sun. Y. Chen, Y. Luo, Edge cloud resource expansion and shrinkage based on workload for minimizing the cost. Future Gener. Comput. Syst. **101**, 327–340 (2019).
41. P. Singh, P. Gupta, K. Jyoti, TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud. Cluster Comput. **22**(4), 619–633 (2019)
42. H.M. Nguyen, G. Kalra, T.J. Jun, S. Woo, D. Kim, ESNemble: an Echo State Network-based ensemble for workload prediction and resource allocation of Web applications in the cloud. J. Supercomput. **75**(10), 6303–6323 (2019)
43. P. Nakaram, T. Leauhatong, A new content-based medical image retrieval system based on wavelet transform and multidimensional wald-wolfowitz runs test. The 5th Biomedical Engineering International Conference (2012).
44. H. Zang, L. Fan, M. Guo, Z. Wei, G. Sun, and L. Zhang, Short-term wind power interval forecasting based on an EEMD-RT-RVM model. Advances in Meteorology, 8760780(2016).
45. J. Chen, Y. Wang, 2018 Sixth International Conference on Advanced Cloud and Big Data. A cloud resource allocation method supporting sudden and urgent requests, pp. 66–70 (2018).
46. B. Tan, H. Ma, Y. Mei, IEEE Congress on Evolutionary Computation (CEC). A NSGA-II-based approach for service resource allocation in cloud **2017**, 2574–2581 (2017)
47. A.S. Sofia, P. GaneshKumar, Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. J. Netw. Syst. Manage. **26**(2), 463–485 (2018)
48. X. Xu, S. Fu, Y. Yuan et al., Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II. Comput. Intell. **35**(3), 476–495 (2019)
49. Alibaba. cluster-trace-v2018. https://github.com/alibaba/clusterdata/-tree/master/cluster-trace-v2018.
50. E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, TIK-report 103, Swiss Federal Institute of Technology (ETH) Zurich (2001).
51. J. Jiang, X. Zhang, S. Li, A task offloading method with edge for 5G-envisioned cyber-physical-social systems. Secur. Commun. Netw., 8867094 (2020).
52. X. Xu, X. Liu, X. Yin, Privacy-aware offloading for training tasks of generative adversarial network in edge computing. Inf. Sci. **532**, 1–15 (2020)
53. G. Rachana, N. S. Jagannath, S. Urvashi Prakash, Cloud detection in satellite images using multi-objective social spider optimization. Appl. Soft Comput. **79**, 203–226 (2019).
54. J. Yang, H. Zhu, T. Liu, Secure and economical multi-cloud storage policy with NSGA-II-C. Appl. Soft Comput. **83**, 105649 (2019)

Publisher's Note