Chen *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2021) 10:20
https://doi.org/10.1186/s13677-021-00237-7

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

A multi-objective optimization for resource allocation of emergent demands in cloud computing

Jing Chen*, Tiantian Du and Gongyi Xiao

Abstract

Cloud resource demands, especially some unclear and emergent resource demands, are growing rapidly with the development of cloud computing, big data and artificial intelligence. The traditional cloud resource allocation methods do not support the emergent mode in guaranteeing the timeliness and optimization of resource allocation. This paper proposes a resource allocation algorithm for emergent demands in cloud computing. After building the priority of resource allocation and the matching distances of resource performance and resource proportion to respond to emergent resource demands, a multi-objective optimization model of cloud resource allocation is established based on the minimum number of the physical servers used and the minimum matching distances of resource performance and resource proportion. Then, an improved evolutionary algorithm, RAA-PI-NSGAII, is presented to solve the multi-objective optimization model, which not only improves the quality and distribution uniformity of the solution set but also accelerates the solving speed. The experimental results show that our algorithm can not only allocate resources quickly and optimally for emergent demands but also balance the utilization of all kinds of resources.

Keywords: Cloud computing, Emergent demands, Resource allocation, Multi-objective optimization, Resource proportion matching distance, Resource performance matching distance

Introduction

Cloud computing applies virtualization technology to divide massive physical resources into various virtual resources [1]. A large number of users can use these virtual resources on a cloud platform anytime and anywhere [2]. More and more applications are being deployed on cloud platforms, whose resource scales have become increasingly large with the application and development of cloud computing, big data and artificial intelligence. Cloud resource demands submitted to them also appear the characteristics of diversity, burst and emergency. Most existing methods of cloud resource allocation do not support the emergent mode, meaning that they cannot guarantee the timeliness and optimization of resource allocation. However, users pay more attention to the timeliness and optimization of their emergent resource demands, and cloud service providers are highly concerned with how to manage massive resources and improve resource utilization. An efficient resource allocation method is crucial to meet these goals. The process of resource allocation is a problem of virtual machine placement for finding the suitable physical servers upon which to place virtual machines (VMs). This process can not only satisfy resource demands of VMs but also improve the resource utilization of the cloud platform. To Find the optimal solution is also a problem. Some simple heuristic algorithms, such as Round Robin (RR) [3], Best Fit (BF) [4], and Min-Max

* 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, China

[5], are applied in the resource allocation process of the small-scale cloud platforms. These algorithms are simple and easy, but they are also prone to waste resources, especially in large-scale cloud platforms.

The bin packing problem is one classical method of VM placement. The VM placement problem on physical servers is transformed into the bin packing problem of n objects packed 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 the empty box [6]. An enhanced variable-sized packing particle swarm optimization algorithm (PSOLBP) is proposed to minimize the number of physical servers and to realize the load balance of physical servers, in which a levy flight algorithm is combined with a particle swarm optimization algorithm (PSO) to avoid the loss of particle dispersion when finding the local and global optimal solutions [7]. An online vector bin packing is used to build a multi-dimensional cloud resource dynamic allocation model (MDCRA), which uses a single weight algorithm and a double weight algorithm to maximize resource utilization, minimize the number of physical servers and energy [8]. The bin packing problem is classified as an NP-hard problem, which only solves the constraints of resource capacity and does not consider incompatible constraints.

Another classical method of cloud resource allocation is to model VM placement as a multi-objective optimization mathematical problem [9–11]. The main idea is to express a cloud resource allocation problem as a multi-objective mathematical function, and then to use a multi-objective evolutionary algorithm to solve it. An optimal VM placement method is proposed, which regards the minimum number of physical servers and the minimum times of VM migration as two objectives and uses a non-dominated sorting genetic algorithm with elite strategy (NSGA-II) to solve this multi-objective problem [12]. A task-oriented multi-objective scheduling method is presented, which takes into account multiple objectives, such as the completion time of tasks, cost and resource utilization, and uses the multi-objective ant colony optimization algorithm (MOSACO) to solve them [13]. A three-dimensional virtual resource scheduling method reduces energy consumption and minimizes service level agreement (SLA) violations from three aspects of virtual resource allocation, scheduling and optimization [14].

The cloud resource allocation is involved in cost-driven [15, 16], energy-saving [17, 18], profit-driven [19], quality of service (QoS) assurance [20, 21], utilization-improving [22, 23] and load balance methods [24, 25]. A systematic resource allocation method based on the random optimization and Lyapunov optimization theories is proposed to ensure users' quality of experience (QoE) and minimize the cost of rented VMs. Considering the trade-off between resource utilization and application performance, a resource allocation algorithm is proposed to meet users' QoS requirements and maximize the resource utilization of a cloud platform [22]. A VM integration scheduling algorithm is proposed based on an active workload-prediction technology and a passive control technology, which uses an exponential smoothing method to predict future workload [26]. There also exist some cloud resource allocation methods for big data applications [27–29], scientific applications [30, 31], video streaming [32], cloud manufacturing [33], mobile applications [34], and workflow [35, 36].

Although some methods have been proposed for saving cost and energy, improving resource utilization and guaranteeing QoS in cloud computing, it is seldom studied how to allocate cloud resources quickly and optimally for emergent demands. A concept of task relaxation degree is proposed based on the deadline time and the execution time of a task, and the task whose relaxation degree is less than a threshold is regarded as the emergent task [37]. And a randomness aware scheduling method is proposed, in which the emergent tasks are executed preferentially on the existing free VMs or the newly added VMs. However, this method only sets some simple priority levels of tasks and does not consider the heterogeneous and diverse emergent resource demands. A machine startup-time-aware strategy is proposed to meet emergent tasks, where an emergent VM is provided and can scale up its CPU capacity dynamically [38]. The method of preparing VMs in advance is only suitable for a small amount of emergent resource demands. If a large number of VMs are prepared in advance, it can cause resource waste due to the idle time. In this paper, the emergent demand denotes the emergent resource demand, which is different from the common resource demand without the requirement of the deadline time of resource provision. The emergent demand should be allocated resources more preferentially than the common resource demand.

Although these methods are effective, most of them allocate resources fairly and do not preferentially satisfy the emergent resource demands. Using these methods will lead to poor effectiveness. An effective cloud resource allocation method should meet the following conditions for emergent resource demands. First, the emergent cloud resource demands should be satisfied preferentially. Second, the optimal cloud resources should be provided. Third, the physical servers should be used as little as possible, and the proportion between different types of resources (number of CPU cores: memory capacity: disk size) should be as uniform as possible to reduce resource waste and improve resource utilization.

To solve the above problems, this paper proposes a cloud resource allocation algorithm for emergent demands. The main contributions of this paper are as follows.

(1) We propose a novel priority of resource allocation based on users' priorities and emergent grades of resource demands, which guarantees the emergent resource demand to be allocated resources preferentially.
(2) We propose a model of resource performance matching, which selects more suitable physical servers to provide resources with better performance for VM requests based on resource performance matching distance between VMs and physical servers.
(3) We propose a model of resource proportion matching, which builds the resource proportion matching distance to ensure the balanced utilization among different types of resources of physical servers.
(4) We propose a method RAA-PI-NSGAII to enhance the quality and distribution uniformity of the solution set and accelerate the solving speed by improving NSGA-II algorithm, which further ensure the timeliness and optimization of cloud resource allocation.

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

Background

Cloud resource allocation process

Resource allocation and scheduling of a cloud platform generally consists of three steps, as shown in Fig. 1. The cloud platform allocates resources to create the requested VM on an appropriate physical server when a user applies for a VM. The cloud platform can scale out or scale in some VMs to ensure the performance of a virtual cluster or reduce the cost according to an auto-scaling strategy when the resource load of a running virtual cluster is too high or too low. One or multiple VMs can also be dynamically migrated to other physical servers to maintain the load balance of the whole cloud platform when a physical server is overloaded.

The first two forms involve in VM creation, which mainly selects the appropriate physical servers to allocate resources for the requested VMs. The final form involves VM migration, which selects virtual machines and migrates them to other appropriate physical servers. This paper mainly studies cloud resource allocation method for the first two forms.

Non-dominated sorting genetic algorithm with elite strategy (NSGA-II)

The key of cloud resource allocation is to find the suitable physical servers to create VMs and improve resource utilization of a cloud platform. Especially, the timeliness and optimization of cloud resource allocation are more important for emergent demands. However, it is not feasible to find the optimal solution by enumerating all feasible solutions because there are a large number of heterogeneous physical servers and various VM requests in a cloud platform. The simple Round Robin (RR) algorithm uses a polling mechanism to select a suitable physical server to create a VM, and Best Fit (BF) algorithm deploys a VM to a physical server that satisfies the VM demand and has the least idle resource. However, these methods generally cause resource waste and resource mismatch because of randomness.

The NSGA-II is an effective multi-objective genetic algorithm [39] often used in solving the multi-objective problem of cloud resource allocation. Figure 2 shows the implementation process of NSGA-II algorithm. First, the

Table 1 List of mathematical notations

Symbol	Annotation	Symbol	Annotation
f^k_{i+1}	the objective value of the *i + 1* the individual on the dimension *k*	v_{ik}	the resource demand of the VM v_i for *k* type of resource
L_i	the priority of a user's resource allocation	p^f_{jk}	The free *k* type of resource of the physical server p_j
v_i	the *i th* VM request	N	the initial population size
p_j	a physical server	x_{ij}	the mapping element between a VM and a PM
MD_{ij}	resource performance matching distance	MP_{ij}	resource proportion matching distance
ε	the threshold of the distance between two individuals	d_{ij}	Euclidean distance of two adjacent *i* and *j* individuals

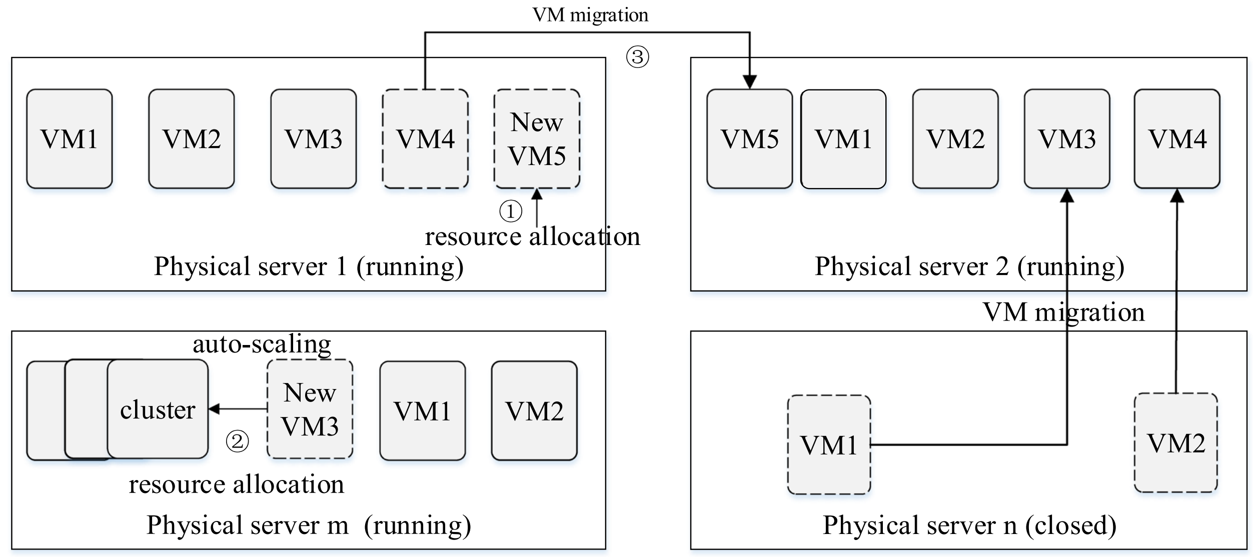

Fig. 1 Resource allocation and scheduling process of a cloud platform

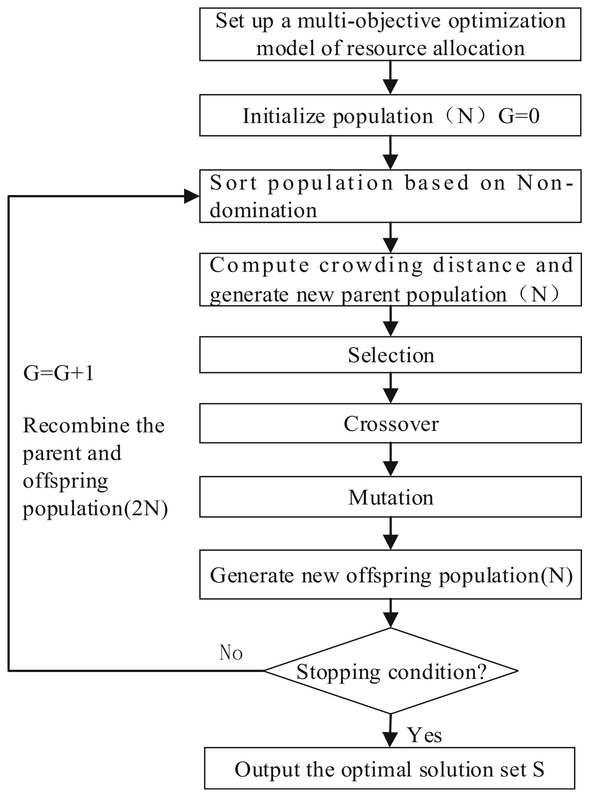
Set up a multi-objective optimization model of resource allocation
Initialize population（N）G=0
Sort population based on Non-domination
Compute crowding distance and generate new parent population（N）
Selection
Crossover
Mutation
Generate new offspring population(N)
Stopping condition?
No
Yes
Output the optimal solution set S
G=G+1
Recombine the parent and offspring population(2N)

Fig. 2 The process of NSGA-II algorithm

population of size N and the iteration times G are initialized. Second, the initialized population is sorted using a non-domination, and all individuals of the population are assigned to a different non-dominant rank. Third, the objective values and the crowding distances of all individuals are calculated according to the following formula (1), where f_{i+1}^k, f_{i-1}^k, represents the objective values of the adjacent $i-1$ and $i+1$ individuals of the i individual on the dimension k.

$$d_i = \sum_{k=1}^{m} | f_{i+1}^k - f_{i-1}^k | \tag{1}$$

Then, the excellent individuals are selected to construct a new parent population according to their non-dominant ranks and crowding distances. If the non-dominant rank of an individual is smaller than that of others, this individual will be selected. The individual with a higher crowding distance is selected if both the individuals have the same non-dominant rank. Then, the parent population goes through selection, crossover and mutation to generate the offspring population of size N. Subsequently, a population of size $2N$ is generated by combining the offspring population with the parent population. Then, the above process continues to be executed until the stopping condition is reached for this combined population.

A multi-objective optimization model for cloud resource allocation

We design the priority of resource allocation to respond to emergent demands, build the matching distance of resource performance to realize resource optimization, and establish a matching distance of resource proportion to balance the utilization of all types of resources. Finally, we consider the benefits of cloud users and service providers and set up a multi-objective optimization model for cloud resource allocation with the goals of minimizing the number of the used physical servers, improving resource utilization and reducing resource fragmentation of the cloud platform.

Priority of resource allocation

A cloud platform generally sets users' priorities to sequentially allocate resources, which cannot respond to emergent demands quickly. Thus, we further propose another parameter: the emergent grade, which indicates an emergent degree of a user's resource demand. Thus, the priority of a user's resource allocation can be calculated by a user's priority and his emergent grade, which guarantees that the emergent resource demands can be satisfied preferentially. However, a user's priority and emergent grade need to be normalized before carrying out the calculation because their units are inconsistent. Supposing the priority of a cloud user is denoted as $R = (R_1, ..., R_i, ..., R_n)$, the normalized user priority can be calculated by formula (2), where $R_{\min}$ and $R_{\max}$ represent the minimum and the maximum value of all users' priorities, respectively, where R_i represents the priority of the i th user. All users' priorities will be set as zero if users' priorities need not be considered in the cloud resource allocation.

$$R_{ni} = \frac{R_i - R_{\min}}{R_{\max} - R_{\min}} \tag{2}$$

Similarly, supposing the emergent grade of users' resource demands is denoted as $G = (G_1, ..., G_i, ..., G_n)$, the normalized emergent grade can be calculated by formula (3), where $G_{\min}$ and $G_{\max}$ represent the minimum and the maximum values of the emergent grades of all users' resource demands, respectively, where G_i represents the emergent grade of the resource demand of the i th user.

$$G_{ni} = \frac{G_i - G_{\min}}{G_{\max} - G_{\min}} \tag{3}$$

Thus, the priority of a user's resource allocation can be calculated using the weighted average method according to formula (4), where α and β represent the weights of the normalized priority R_{ni} and the emergent grade G_{ni}, respectively.

$$L_i = \alpha \cdot R_{ni} + \beta \cdot G_{ni} \qquad (\alpha + \beta = 1) \tag{4}$$

Resource performance matching distance

To meet the emergent resource demands, it is necessary to match the optimal resources in addition to ensuring the timeliness of resource allocation, which means that a VM should be placed on a physical server, whose resource performance is more appropriate for the performance requirement of the VM. For instance, if a user requests a compute-intensive virtual machine, a compute-intensive physical server will be selected to allocate resources for this VM. Thus, we set up a resource performance matching distance between a VM and a physical server based on their resource performance vectors. The smaller the resource performance matching distance, the better the resource performance matching between them.

Users' resource demands are expressed as a request queue $V = < v_1, ..., v_i, ..., v_n >$, where v_i represents the i th VM and n denotes the sequence number of VM requests. The resource demand of the VM v_i is expressed as $v_i(v_{ic}, v_{im}, v_{id})$, where v_{ic}, v_{im} and v_{id} denote the number of CPU cores, the memory capacity and the disk size requested by the VM v_i, respectively. The available physical servers of a cloud platform are represented as a physical server group $P = (p_1, ..., p_j, ...p_n)$. Each physical server has the total resource capacity $p_j = (p_{jc}, p_{jm}, p_{jd})$ and the free resource capacity $p_j(p^f_{jc}, p^f_{jm}, p^f_{jd})$, where p_{jc} and p^f_{jc}, p_{jm} and p^f_{jm}, p_{jd} and p^f_{jd} represent the total and free number of CPU cores, memory capacities and disk sizes, respectively.

The performance vector of a VM v_i is expressed as $vv_i = (vv_{ic}, vv_{im}, vv_{id})$ for a physical server p_j. The elements in this vector can be calculated by the formulas.

$$\text{CPU performance vector} : vv_{ic} = v_{ic}/p_{jc} \tag{5}$$

$$\text{Memory performance vector} : vv_{im} = v_{im}/p_{jm} \tag{6}$$

$$\text{Disk performance vector} : vv_{id} = v_{id}/p_{jd} \tag{7}$$

Similarly, the performance vector of a physical server is denoted as $pv_j = (pv_{jc}, pv_{jm}, pv_{jd})$. The elements in this vector can be calculated by the formula as follows.

$$\text{CPU performance vector} : pv_{jc} = p^f_{jc}/p_{jc} \tag{8}$$

$$\text{Memory performance vector} : pv_{jm} = p^f_{jm}/p_{jm} \tag{9}$$

$$\text{Disk performance vector} : pv_{jd} = p^f_{jd}/p_{jd} \tag{10}$$

Then, we calculate the normalized performance vector $nvv_i = (nvv_{ic}, nvv_{im}, nvv_{id})$ of the VM v_i and the normalized performance vector $npv_j = (npv_{jc}, npv_{jm}, npv_{jd})$ of the physical server p_j according to their performance vectors vv_i and npv_j. Consequently, the optimal objective function—that is, the resource performance matching distance MD_{ij} between the VM v_i and the physical server p_j—is established on their performance vectors. The smaller the distance, the better will be the resource performance matching between the VM and the physical server.

$$MD_{ij} = \sqrt{\sum_{k\in\{c,m,d\}} \left(npv_{ik} - npp_{jk}\right)^2} \tag{11}$$

And the resource demands of all VMs placed on a physical server should be less than the free resource of this physical server p_j. So the constraint should be considered as follows.

$$\sum_{i=1}^{m} v_{ik} < p_{jk}^{f} \quad (k = c, m, d; j = 1, 2, ..., n) \tag{12}$$

Resource proportion matching distance

Physical servers are heterogeneous, and VM demands are different for all kinds of resources in cloud computing. When a VM is created on a physical server, it can lead to a varied proportion among different types of free resources of the physical server, such as the free number of CPU cores, memory capacity and disk size. When one type of resource has been exhausted, a physical server will no longer create a new VM despite there still remaining a large amount of other types of resources. For instance, if the CPU resource of a physical sever is exhausted, it cannot create VMs though it may have large memory capacity and disk size, which will result in resource fragment generation and resource waste. To reduce resource waste, it is necessary to consider the proportion between different types of resource demands of VMs and different types of free resources of physical servers. A VM should be created on the physical server whose resource proportion is closer to that of this VM. Thus, the proportion of different types of resources can always maintain uniformity on a physical server, which reduces the probability of resource fragment generation. Therefore, the resource allocation problem changes into a problem of finding a resource proportion reflection of the resource demands of VMs and free resources of physical servers. A physical server should be selected to allocate resources for this VM because its proportion among different types of free resources is closer to that among different types of requested resources of this VM, that is, if $p_{jc} : p_{jm} : p_{jd}$ is closer to $v_{ic} : v_{im} : v_{id}$, where v_{ic}, v_{im} and v_{id} denote the requested number of CPU cores, memory capacity and disk size of the VM v_i and p_{jc}, p_{jm} and p_{jd} denote the free number of CPU cores, memory capacity and disk size of the physical server p_j, the physical server p_j should be selected to allocate resources for the VM v_i. The resource proportion matching distance between different types of resources is set up between a VM and a physical server by the following formula. The closer the resource proportion matching distance is, the fewer resource fragments and the greater the resource utilization there will be.

$$MP_{ij} = \sqrt{\sum_{k\in\{c,m,d\}} \left(\frac{p_{jk}^{f} \cdot v_{ic}}{p_{jc}^{f}} - v_{ik}\right)^2} \tag{13}$$

Similarly, the constraint should be considered as follows.

$$\sum_{i=1}^{m} v_{ik} < p_{jk}^{f} \quad (k = c, m, d). \tag{14}$$

A multi-objective optimization model of resource allocation

There are a large number of physical servers in a cloud platform, which generates a large number of VMs to provide resource services. These VMs always manifest with different performances regardless of whether the physical servers are heterogeneous or homogeneous. A good algorithm should ensure the fastest and optimal resource allocation for emergent demands. The process of resource allocation is to place a virtual machine to the appropriate physical server. If a VM v_i is placed on a physical server p_j, the mapping element x_{ij} between this VM and this physical server should satisfy $x_{ij} = 1$; otherwise, $x_{ij} = 0$. The mapping matrix from the VM request queue V to the physical server group P may be expressed as follows.

$$X = \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{pmatrix} \tag{15}$$

The mapping from the VM queue to the physical server group can be expressed as follows.

$$(v_1, ..., v_i, ..., v_m) \times \begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{pmatrix} = \begin{pmatrix} p_1 \\ \dots \\ p_j \\ \dots \\ p_n \end{pmatrix} \tag{16}$$

So, the number of the used physical servers can be calculated by summing all elements of the matrix X—that is, $\sum_{i=1}^{m} x_{ij}$. Thus, a multi-objective optimal resource allocation model can be established based on the minimum number of the used physical servers, the minimum resource performance matching distance and the minimum resource proportion matching distance, as follows.

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}\right\} \tag{17}$$

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} MD_{ij}\right\} \tag{18}$$

$$\min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} MP_{ij}\right\} \tag{19}$$

The constraints should satisfy the relation that the resource demands of all VMs placed on a physical server p_j cannot exceed its free resources as follows.

$$S.T. \sum_{i=1}^{m} v_{ic} \cdot x_{ij} \le p_{jc}^{f} \ (j = 1, 2, ..., n) \tag{20}$$

$$\sum_{i=1}^{m} v_{im} \cdot x_{ij} \le p_{jm}^{f} \quad (j = 1, 2, ..., n) \tag{21}$$

$$\sum_{i=1}^{m} v_{id} \cdot x_{ij} \le p_{jd}^{f} \quad (j = 1, 2, ..., n) \tag{22}$$

And if one type of free resource capacity p_{jh}^{f} of a physical server p_j cannot satisfy the resource demand v_{ih} of any VM, the ratio of other type of free resource capacity p_{jk}^{f} to the total resource capacity p_{jk} should be less than a threshold ε_k as the formula (23), where the symbol h denotes one type of resource, such as CPU, and the sign k denotes another type of resource, such as memory or disk. Thus, three types of resources will be used evenly, which largely reduces resource fragments.

$$if\, \exists v_{ih} > p_{jh}^{f}, then\, \frac{p_{jk}^{f}}{p_{jk}} \le \varepsilon_k \ (h \ne k, j = 1, 2, ..., n) \tag{23}$$

Thus, the resource allocation algorithm is transformed into the solution of a multi-objective mathematical model. This multi-objective mathematical problem is NP-hard because its solution is not uniquely definite value, that is, it is not single but multiple. These solutions can be obtained by using a multi-objective evolutionary algorithm, but they cannot be compared.

The multi-objective optimization algorithm RAA-PI-NSGAII

A multi-objective optimization mathematical problem can be solved by a multi-objective evolutionary algorithm. Moreover, the algorithm should accelerate the solution process and improve the quality of the solution set to ensure the timeliness and optimization of resource allocation for emergent demands. NSGA-II is a non-dominated sorting genetic algorithm that has been used to solve the multi-objective optimization problems and has achieved good effectiveness [40–44].

The traditional NSGA-II algorithm has three problems in solving the multi-objective optimization model of resource allocation, as follows. First, the computation time of the values of objective functions is too long to allocate resources in a timely manner; hence, it cannot meet emergent resource demands. Second, the individuals from the parent and offspring populations may be repetitive in the process of population evolution. After the parent and offspring individuals are merged, the repetitive individuals have non-dominant relationships and are assigned the same hierarchical rank. These individuals may be selected into the next-generation population because their crowding distances may be larger than those of the non-repetitive individuals, which will cause many repetitive solutions to enter into the optimal solution set. As shown in Fig. 3, the *a* and *b* points are the repetitive individuals, and their crowding distances are greater than the points *h* and *i*; consequently, they are preferentially selected to enter the next-

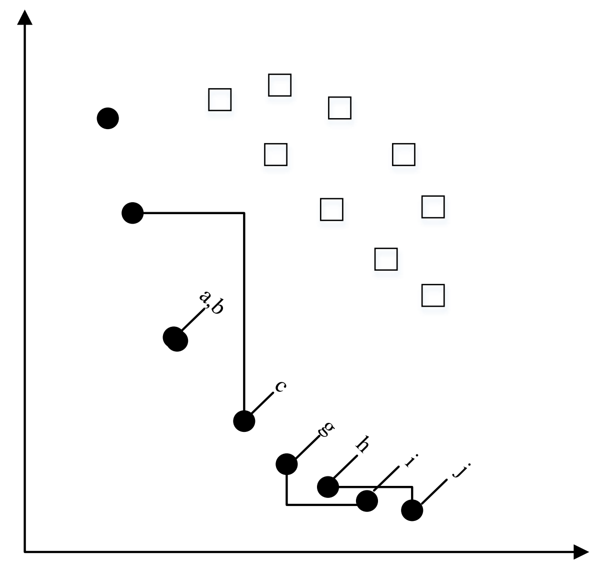

Fig. 3 Repetitive individual problem

generation population. Third, the distribution of the solution set is not uniform. As shown in Fig. 4, although the a individual is very close to the b individual, both of them may be retained because of their larger crowding distances. In contrast, both the f individual and the g individual may be eliminated, which results in an uneven population distribution. Obviously, the c individual and the d individual have the same situation, and a good selection is to retain one of them.

Therefore, we propose a cloud resource allocation algorithm based on a parallel and improved NSGA-II algorithm (RAA-PI-NSGAII), which computes the fitness values of individuals concurrently, removes the repetitive individuals and selects the adjacent excellent individuals to improve the quality of the solution set, accelerates the solving speed, and optimizes the distribution uniformity of the solution set. Thus, the timeliness and optimization of resource allocation are further guaranteed for emergent demands.

The multi-objective optimization algorithm of cloud resource allocation RAA-PI-NSGAII improves the NSGA-II algorithm in the following respects.

Parallel computation and evaluation of the fitness function

The computation and evaluation time of the fitness values is very long due to too many individuals in the population. The multi-core processors calculate the fitness values of individuals and evaluate them in parallel, which can speed up the convergence rate of the proposed algorithm. We calculate and evaluate the fitness values (i.e., objective functions) of each individual as follows.

$$f1 = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \tag{24}$$

$$f2 = \sum_{i=1}^{m}\sum_{j=1}^{n} MD_{ij} \tag{25}$$

$$f3 = \sum_{i=1}^{m}\sum_{j=1}^{n} MP_{ij} \tag{26}$$

Optimal selection of adjacent individuals

Some individuals remain after all individuals are sorted in a non-dominant order, and the repetitive individuals who cause the non-uniformity distribution of the solution set are removed. Consequently, we should further select excellent individuals to enter the next generation population. We calculate the Euclidean distance between two adjacent individuals to determine whether the distance is less than a threshold. If it is, the excellent individual is selected according to the following strategy.

Definition of threshold

Calculate the maximum Euclidean distance $\max(D_{rank(i)})$ of two individuals of the non-dominant set and the threshold ε according to formula (27), where N is the population size.

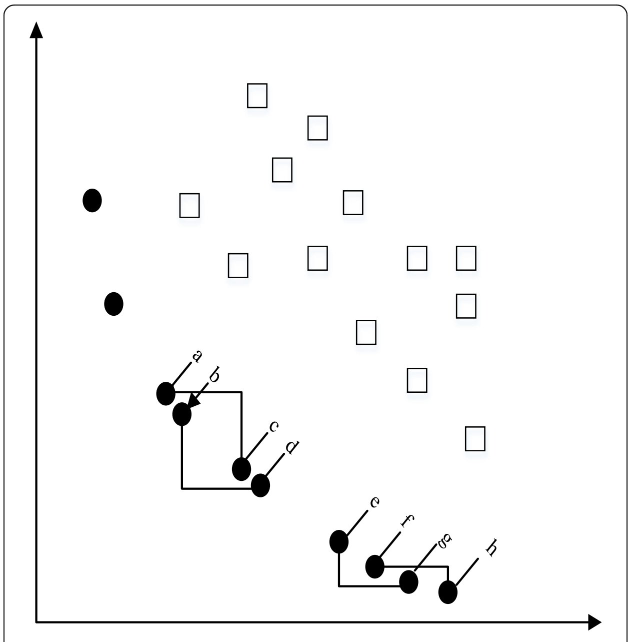

Fig. 4 Population distribution problem

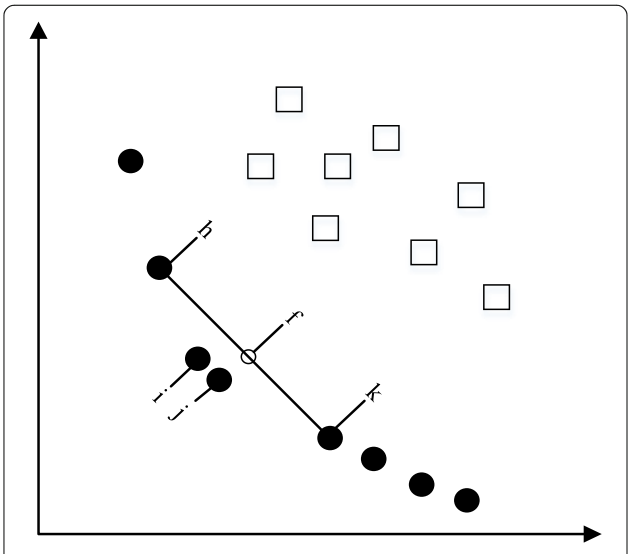

Fig. 5 Selection of adjacent excellent individuals

$$\varepsilon = \frac{\max\left(D_{rank(i)}\right)}{2 \times N} \tag{27}$$

Selection of adjacent excellent individuals

The Euclidean distance d_{ij} of two adjacent i and j individuals is first calculated after removing repetition in the non-dominant set. As shown in Fig. 5, if the distance d_{ij} is less than the threshold ε, the individuals h and k adjacent to them are determined, and their centre point f is calculated. Then, the individual from the set of i and j that is closer to f is retained. As a result, the individual j is retained, and the individual i is removed.

The implementation of the whole algorithm consists of two parts. The parameters are initialized, and the multi-objective optimization function of resource allocation is established in Algorithm 1. The improved NSGA-II algorithm is used to solve the multi-objective optimization function, as shown in Algorithm 2. The main steps of the implementation of the algorithm RAA-PI-NSGAII are as follows.

- Step 1: Construct a multi-objective optimization resource allocation model.
- Step 2: Initialize the parameters of RAA-PI-NSGAII algorithm, such as the parent population S_1, the offspring population Q_1, crossover and mutation probability P_c, cross distribution index I_{cd}, mutation probability P_m, mutation distribution index I_{md}, maximum iteration times G and population size N.
- Step 3: Merge the parent population S_i and the offspring population Q_i.
- Step 4: The fitness values of the individuals from the merged population R_m are calculated in parallel, and the improved NSGA-II algorithm is used to carry out non-dominant sorting.
- Step 5: Remove the repetitive individuals on each non-dominant set and select the excellent one from the adjacent individuals.
- Step 6: If the accumulative number of individuals selected from k non-dominant sets is greater than the number N, then the individuals in the k th non-dominant set are sorted from the largest crowding distance to the smallest one, and a new parent population R_{m+1} is formed with the N selected individuals.
- Step 7: Select, cross and mutate the individuals from the new parent generation population R_{m+1} to generate the new offspring population.
- Step 8: Determine whether the stopping conditions have been reached or not. If they are reached, the Pareto optimal solution set of resource allocation is output. If it is not, step 3 is continued to be executed until the stopping conditions is reached.

```
Algorithm 1: Multi-objective optimization resource allocation function Objective(V,U,H)
Input: VM queue V, emergent grade of VM requests H, server set P
1.  Initialize users' priorities U, number of VM requests $N_v$, number of servers $N_h$;
2   $(U_{n1},...,U_{ni},...,U_{nm}) \leftarrow normalize(U)$;
3   $(H_{n1},...,H_{ni},...,H_{nn}) \leftarrow normalize(H)$;
3   $A_i = \alpha \cdot U_{ni} + \beta \cdot H_{ni}$;          //calculating the priorities of resource allocation
4   for j=1 to $N_h$ do
5       $pp_{jc} \leftarrow p^f_{jc} / p_{jc}$;
6       $npp_{jc} \leftarrow normalize(pp_{jc})$;
7       similarly, calculate the normalized memory and disk performance vector $npp_{jm}, npp_{jd}$;
8   end for
9   for i=1 to $N_v$ do
10      similarly, calculate the normalized CPU, memory and disk performance vectors
        $npv_{ic}, npv_{im}, npv_{id}$;
11      $j = s(i)$;
12      $MD_{ij} \leftarrow \sqrt{\sum_{k \in \{c,m,d\}} (npv_{ik} - npp_{jk})^2}$;  //resource performance matching model
13      $MP_{ij} \leftarrow \sqrt{\sum_{k \in \{c,m,d\}} \left( \frac{p^f_{jk} \cdot v_{ic}}{p^f_{jc}} - v_{ik} \right)^2}$;  //resource proportion matching model
14      if $v_{ic} < p^f_{jc}$ & $v_{im} < p^f_{jm}$ & $v_{id} < p^f_{jd}$
15          $x_{ij} = 1$;
16          $objective1 = objective1 + MD_{ij}$;
17          $objective2 = objective2 + MP_{ij}$;
18      end if
19  end for
20  for j=1 to $N_h$ do
21      $objective3 = objective3 + x_{ij}$;
22  end for
```

```
Algorithm 2: The improved NSGA-II algorithm
Input: VM queue V, emergent grade of VM requests H, server set P
Output: Pareto optimal solution set S
1.   Initialize $S_1$, $Q_1$, $P_c$, $I_{cd}$, $P_m$, $I_{md}$, $G$, $N$
2.   for m=1 to G do
3.       $R_m \leftarrow S_m + Q_m$;
4.       $F \leftarrow$ fast_nondominated_sort( $R_m$ )
5.       while $|R_{m+1}| + |F_k| \leq N$ do
6.           for any two individuals $t$ and $i$ in $F_k$ non-dominant set do
7.               if $obj_s(t) = obj_s(i)$ then
8.                   remove the duplicate individual $t$, keep $i$ and insert $i \rightarrow R_{m+1}$
9.               else
10.                  $d(i,j) = sqrt(sum((obj_s(i) - obj_s(j))^2))$  //distance between adjacent individuals
11.                  if $d(i,j) < \varepsilon$
12.                      remove $i$ farther to the middle position of $h$ and $g$, remain $j \rightarrow R_{m+1}$
13.                  end if
14.              end if
15.          end for
16.          k=k+1;
17.      end while
18.      $F'_k \leftarrow$ RankingAndCrowdingSelection( $F_k$, $N - |R_{m+1}|$ );
19.      $R_{m+1} \leftarrow R_{m+1} \cup F'_k$
20.      $Q_{m+1} \leftarrow$ SBXCrossover( $P_c$, $I_{cd}$ ), PolynomialMutation( $P_m$, $I_{md}$ );
             BinaryTournamentSelection ( $R_{m+1}$ );
21.      m=m+1;
22.  end for
23   output S
```

In theoretical analysis, the algorithm RAA-PI-NSGAII takes into account users' priorities and emergent grades to calculate the priority of resource allocation, which guarantees the rapid resource allocation for the urgent resource demands. Additionally, the resource performance matching distance depicts the distance between a VM performance vector and a physical server performance vector, which ensures more suitable resource allocation for urgent resource demands. Finally, the resource proportion matching distance makes the proportion among different types of resources of a VM request closer to that among different types of resources of a physical server, which reduces the resource fragment of a physical server.

Experimental setup and metrics

Experimental setup

We use the cloud computing simulation software 'CloudSim' and a multi-objective algorithm framework 'Jmetal' to test and verify the effectiveness, repetitiveness removal, distribution uniformity and solving time of our algorithm. The simulation program is written in Java and deployed on a Lenovo Thinkpad notebook (8G memory, Intel Core i7 6500U).

Three types of resources (CPU, memory and disk) are considered in this experiment. A cloud datacenter composed of 400 physical servers is simulated using 'Cloudsim' software. There are four types of physical servers, and each type includes 100 physical servers. Table 2 shows the total and free number of CPU cores, memory capacity and disk size for each type of physical server. There exist five common specifications of VMs in Table 3. In this experiment, we assume five users applying for different types but having the same number of VMs. Users' priorities are positive integer numbers generated randomly within integer number set [1, 5], with larger number corresponding to higher priorities. We use the emergent grades to express the degrees of emergent resource demands. The emergent grades of VM requests are also generated randomly within the integer number set [1, 5]. Similarly, the greater the value, the higher the emergent degree is. The total numbers of VMs requested by users are set as the values 100, 200, 400, 600, 800, and 1000, respectively. The units of different types of resources differ, so the direct calculation of the original data will cause the proportion among them to be dominated by one type of resource. For instance, the proportion among free number of CPU cores, free memory capacity and free disk size is 48:192:1800 for a P1 type of physical server, which may cause the matching distance of resource proportion to be dominated by the disk. We divide the data of the disk size by 10 to prevent the domination of disk resource.

Table 2 Type of physical servers

Types of physical servers	P1	P2	P3	P4
Total number of CPU cores	64	32	64	64
Total memory capacity (GB)	256	64	128	128
Total disk size (GB)	2048	1024	2048	1024
Free number of CPU cores	48	28	42	50
Free memory capacity (GB)	192	56	86	82
Free disk size (GB)	1800	800	1700	900

Table 3 Type of VMs

Types of VMs	V1	V2	V3	V4	V5
Number of CPU cores	1	2	2	4	8
Memory capacity (GB)	2	2	4	8	8
Disk size (GB)	80	100	120	150	200

The population size, the crossover probability, the crossover distribution index and the mutation distribution index are set as 200, 0.85, 20 and 20, respectively, in the experiment on the RAA-PI-NSGAII algorithm. The mutation probability is set as the reciprocal of the number of variables. The maximum number of evaluations of fitness values and iterations are set as 20,000 and 100, respectively. We execute each algorithm 10 times and compute the average results.

Metrics

To verify the effectiveness and performance of the RAA-PI-NSGAII algorithm, we evaluate and compare it with other algorithms on these following metrics.

Number of the used physical servers

The physical servers should be used as little as possible when VMs are created according to the resource allocation algorithm. The fewer used physical servers there are, the more the idle physical servers can be closed, thereby reducing energy consumption and cost.

Matching distances of resource performance and resource proportion

The matching distances of resource performance and resource proportion can be used to evaluate the matching degree of the resource demands of VMs and the free resources of the physical servers. The smaller the distances, the better the virtual resource demands match with physical resources. Thus, the VMs will have better performance, and fewer resources are wasted.

Resource utilization

A good multi-dimensional resource allocation algorithm should maximize and homogenize each type of resource utilization.

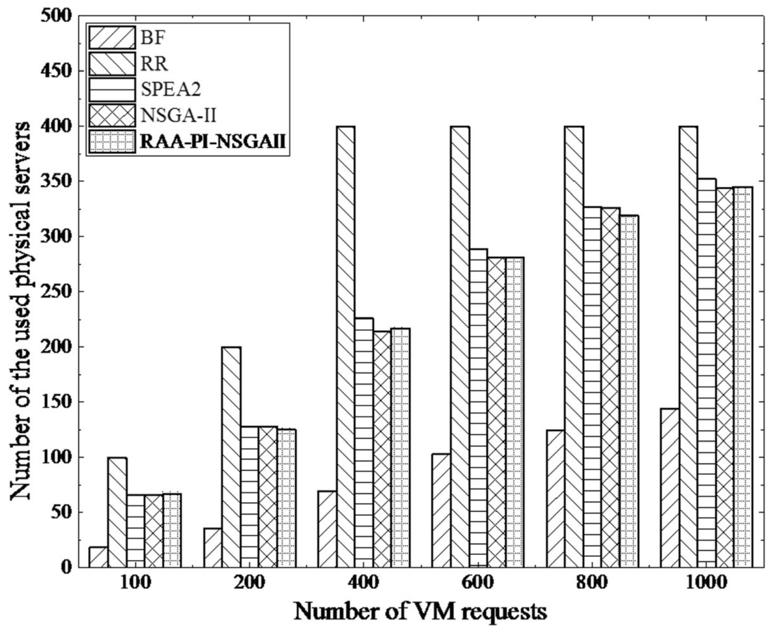

Fig. 6 Number of the used physical servers

Repetition rate of the solution set

The repetition rate of the solution set is defined as the ratio of the repetitive solutions to the total number of solutions. The smaller its value, the better the solution set is, which indicates that the algorithm has better effectiveness with regard to resource allocation.

Distribution uniformity of the solution set

The spacing metric (SP) is an effective index that can be used to evaluate the distribution uniformity of the solution set, as proposed by Schoot. The smaller the SP value, the better the distribution of the solution set is. SP is formulated as follows, where n is the number of solutions, d_k is the distance between the individual k and its nearest individual h, and $\overline{d}$ is the average value of d_k.

$$SP = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}\left(\overline{d}-d_k\right)^2} \tag{28}$$

The distance d_k of the adjacent individuals is defined as follows, where f_l^k is the value of the k th individual on the objective dimension l.

$$d_k = \min\left\{\sum_{l=1}^{m}|\left(f_l^k - f_l^h\right)|\right\} \tag{29}$$

Time cost

The time of VM resource allocation includes the solving time of the RAA-PI-NSGAII algorithm, the VM waiting time in the queue and the VM creation time. The total VM waiting time and creation time of all VMs are

Table 4 Matching distances of resource performance and resource proportion

Number	BF		RR		SPEA2		NSGA-II		RAA-PI-NSGAII	
of VMs	MD	MP	MD	MP	MD	MP	MD	MP	MD	MP
100	116	670	120	1024	60	451	56	402	56	398
200	234	1378	275	1470	142	1036	137	932	132	887
400	467	2661	496	2417	309	2400	298	2177	291	2051
600	700	4061	761	4394	461	3799	457	3532	456	3435
800	871	5464	975	5413	615	5342	614	5010	613	5030
1000	1046	6600	1235	6670	779	6585	763	6446	764	6400

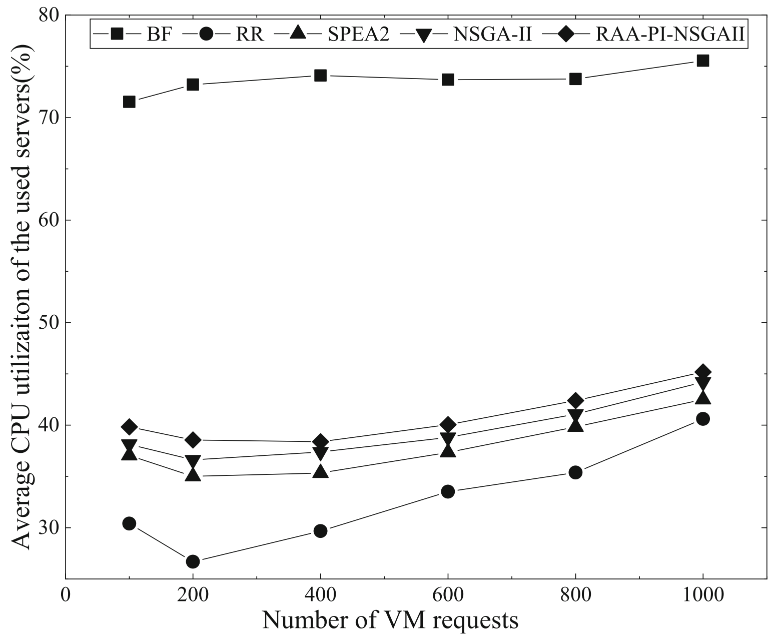

Fig. 7 Average CPU utilization of the used physical servers

similar for different resource allocation algorithms regardless of the influence of the network topology and server load. Therefore, we use the solving time as the algorithm runtime to evaluate the performance of the multi-objective optimization algorithms.

Experimental results and analysis

In the experiments, we recalculate the priorities of resource allocation according to users' priorities and emergent grades generated randomly. For example, 400 VM requests from 5 users have different emergent grades

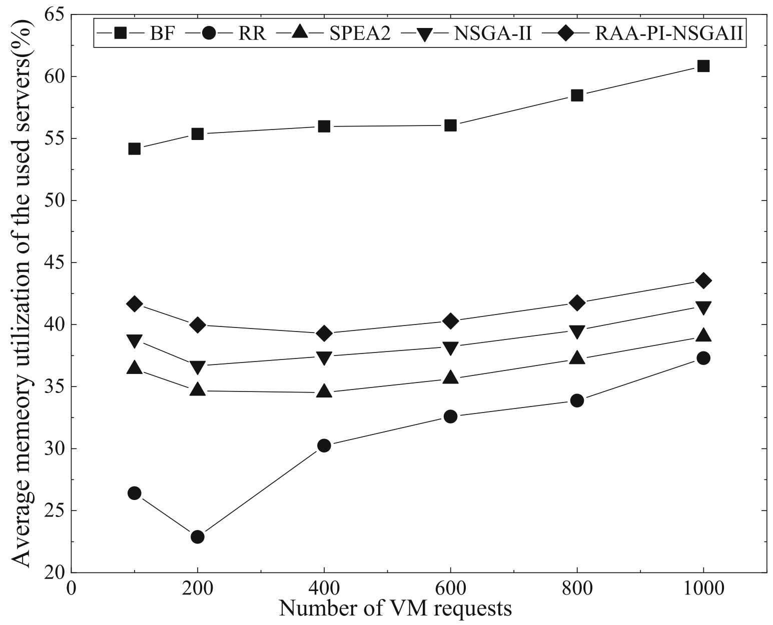

Fig. 8 Average memory utilization of the used physical servers

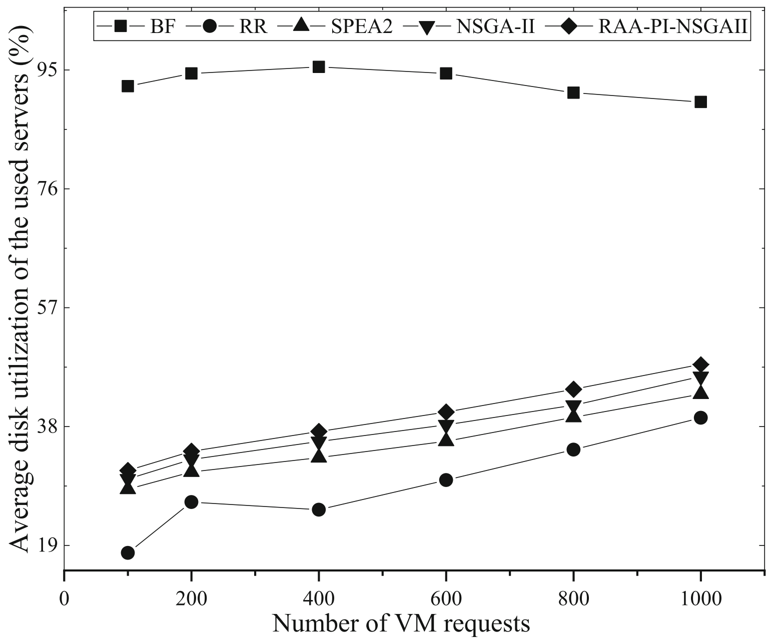

Fig. 9 Average disk utilization of the used physical servers

and users' priorities. The 215th, 219th, 223th, 227th, 230th, 236th, 245th, 264th, 279th and 282th VM requests in the queue belong to the same user with the highest user priority and have the highest emergent grades. These VMs are preferentially allocated resources by the first to the 10th order in the queue, which ensures the timelessness of resource allocation for VMs with higher priorities. To verify the effectiveness of our proposed RAA-PI-NSGAII algorithm, we compare it with the RR, BF, NSGA-II and SPEA2 algorithms according to the number of the used physical servers, resource utilization, resource matching, repetitive solution removal, and solution distribution uniformity. The BF algorithm is a single-type resource allocation method. CPU is the allocated resource type in the BF algorithm experiment.

A cloud platform receives various VM requests and then creates VMs on physical servers according to the solution of a resource allocation algorithm. Figure 6 shows the number of physical servers used for different VM requests. It can be observed that the BF algorithm uses the fewest physical servers. For instance, 600 VMs are created on 103 physical servers and 1000 VMs are created on 144 servers. The BF algorithm first creates a VM on a physical server that can meet the resource demand of this VM and has the fewest free resources. The RR algorithm uses the largest number of physical servers due to its polling mechanism. The multi-objective evolutionary algorithm SPEA2, NSGA-II, and our proposed RAA-PI-NSGAII algorithm use more physical servers than the BF algorithm but fewer than the RR algorithm. Since these three algorithms aim to use the fewest physical servers, the solving results are similar. Each algorithm uses almost the same number of physical servers for the same VM requests. Relatively, the RAA-PI-NSGAII algorithm uses fewer physical servers than the

Table 5 Repetition rate of a solution set

Number of VMs	SPEA2	NSGA-II	RAA-PI-NSGAII
100	27.20%	24.59%	0.00%
200	27.05%	22.36%	1.36%
400	25.00%	0.00%	0.00%
600	24.75%	0.00%	0.00%
800	27.17%	23.49%	0.00%
1000	25.75%	19.91%	0.00%

Table 6 Spacing metric of algorithm distribution

Number of VMs	SPEA2	NSGA-II	RAA-PI-NSGAII
100	1.50	1.94	1.73
200	2.67	4.60	4.01
400	4.36	9.66	7.55
600	6.13	11.71	8.59
800	6.83	13.23	11.80
1000	7.43	14.97	12.85

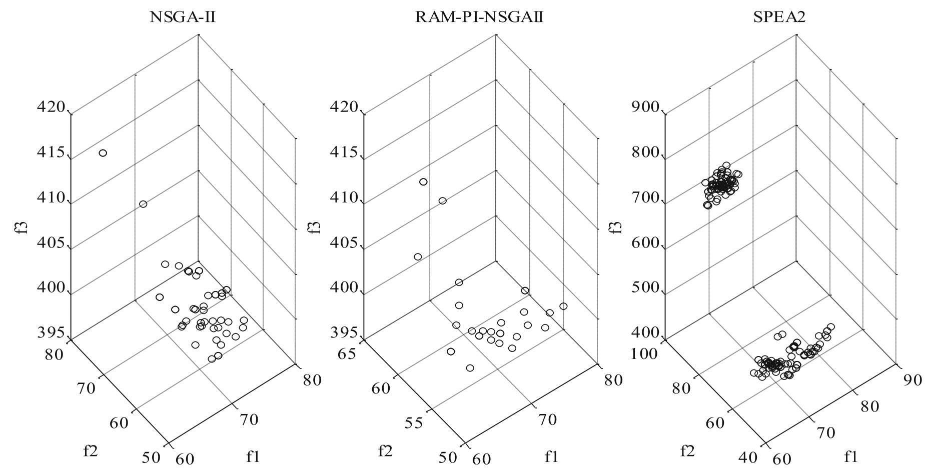

Fig. 10 Pareto optimal solution set for 100 VM requests

SPEA2 and NSGA-II algorithms. The RAA-PI-NSGAII algorithm only uses 319 physical servers for 800 VM requests, while both the SPEA2 and NSGA-II algorithms use over 325 physical servers.

Table 4 shows the experimental results of the different algorithms according to their resource performance and resource proportion matching. It can be seen that the MD and MP of the RR algorithm reach maximum values because it adopts a polling strategy and does not consider any resource matching. The BF algorithm places virtual machines on physical servers with the fewest CPU resources, so its MD value is slightly lower than that of RR algorithm. The multi-objective evolutionary algorithms SPEA2, NSGA-II and RAA-PI-NSGAII aim at achieving the minimum number of physical servers, the minimum matching distance of resource performance and the minimum matching distance of resource proportion, so their MD and MP values are smaller than those of the BF and RR algorithms. Among them, the MD and MP values obtained by the RAA-PI-NSGAII algorithm are smaller than those of the SPEA2 and NSGA-II algorithms, which demonstrates that the RAA-PI-NSGAII algorithm is more effective than other algorithms in regard to matching the optimal physical resources for VMs and reducing resource fragments.

Figures 7, 8, 9 show the effectiveness of different algorithms in regard to improving resource utilization. It is noted that the curves of NSGA-II and RAA-PI-NSGAII methods are offset by 2 units along the Y-axis to make them clear. The BF algorithm deploys VMs on fewer servers, which can cause the resource load to grow too fast to maintain the stability of a cloud platform. The RR algorithm appears to have the lowest utilization of the CPU, memory and disk, but the use of these resources

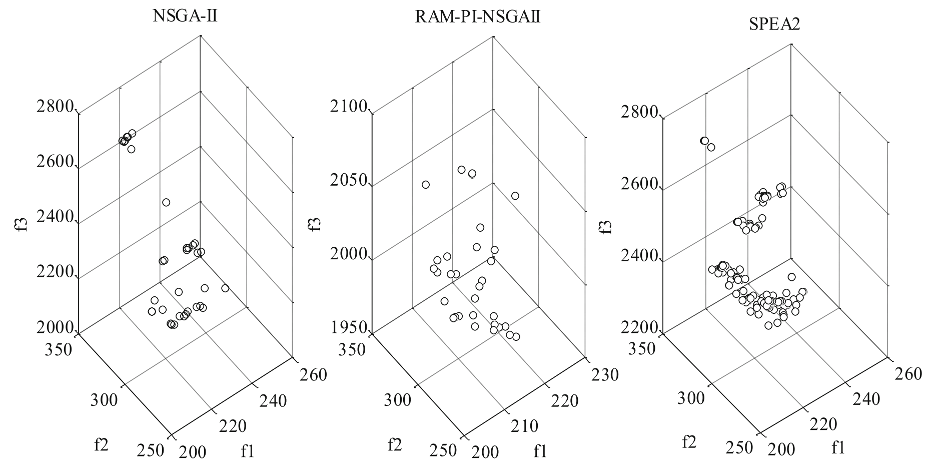

Fig. 11 Pareto optimal solution set for 400 VM requests

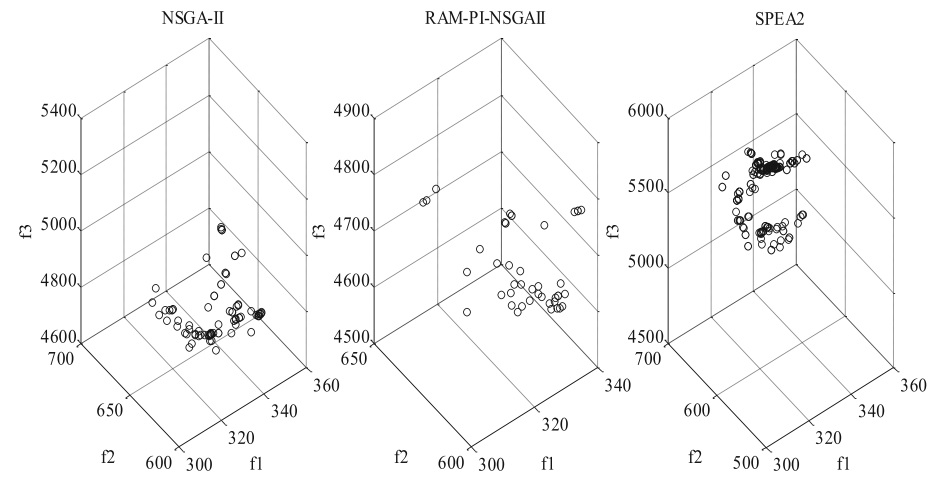

Fig. 12 Pareto optimal solution set for 800 VM requests

increases much faster than that of other algorithms with the increase of the number of VM requests. Because the SPEA2, NSGA-II, and RAA-PI-NSGAII algorithms consider the resource proportion matching of different types of resources, their utilization of the CPU, memory and disk are always stable and slightly increase with the increase of the number of VM requests, which is beneficial to the stability and resource optimization of a cloud platform.

The repetitive individuals not only deteriorate the search efficiency due to the overlapped search space but also greatly influence the evaluation of the solution distribution. The repetition rates of the solution set of the SPEA2, NSGA-II and RAA-PI-NSGAII algorithms are compared in Table 5. The SPEA2 algorithm has the highest repetition rate, which is greater than 20% or even up to 27.20%. The NSGA-II algorithm has the second highest repetition rate, and its repetition rate is close to 25%. Our proposed RAA-PI-NSGAII algorithm greatly reduces the number of repetitive individuals and achieves a 1.36% repetitive ratio.

To validate the advantages of our algorithm, we calculate the SP values of three algorithms in Table 6. The average SP values of the RAA-PI-NSGAII algorithm are smaller than those of the NSGA-II algorithm, which shows that the RAA-PI-NSGAII algorithm improves the distribution uniformity of the solution set. However, we also find that SP values of the RAA-PI-NSGAII algorithm are obviously higher than those of the SPEA2 algorithm.

We randomly select a solution of each algorithm for further analysis. The distribution of the Pareto optimal solution set of the SPEA2, NSGA-II and RAA-PI-NSGAII algorithms are shown in Figs. 10, 11 and 12 for different VM requests. It can be seen that the solution distribution of the NSGA-II algorithm is not uniform. The solution is sparse or even zero in some regions, while the solution is dense and repeated in other regions. We can also observe that the SPEA2 algorithm has many repetitive solutions that stack together. Because the zero distance between the repetitive solutions greatly affects the distribution of the solution set, the SPEA2 algorithm achieves low SP values. The distribution of the solution set of the RAA-PI-NSGAII algorithm. is relatively uniform, but may not be dense, and there are still some sparse regions. This paper mainly focuses on reducing the solving time and improving the quality and distribution uniformity of the solution set. In the future, we will further use Hypervolume (HV) to evaluate the convergence and diversity of the obtained solution set.

We performed an experiment to obtain the solving time of each algorithm. The solving time of the BF and RR algorithms are both short, within 1 s, while those of the multi-objective evolutionary SPEA2, NSGA-II, RAA-PI-NSGAII algorithms are relatively long due to performing multiple generation evolution of the population. Table 7 compares the results of the RAA-PI-NSGAII, SPEA2 and NSGA-II algorithms. The NSGA-II algorithm is more effective than the SPEA2 algorithm. The NSGA-II algorithm takes 5 and 20 s less than the SPEA2 algorithm to perform 200 and 600 VM requests, respectively. We use 8, 32 and 128 threads to

Table 7 Solving time of the multi-objective evolutionary algorithms

Number of VMs	SPEA2(ms)	NSGA-II (ms)	RAA-PI-NSGA-II (ms)		
			8 T	32 T	128 T
200	152,594	147,534	57,064	43,036	40,882
400	292,013	293,395	113,385	83,720	79,987
600	411,650	392,866	159,934	115,199	118,689

solve the multi-objective problem based on the RAA-PI-NSGAII algorithm. Obviously, the solving time of the RAA-PI-NSGAII algorithm is greatly reduced compared to those of the SPEA2 and NSGA-II algorithms. The solving time of the RAA-PI-NSGAII algorithm using 8-thread parallel execution is 40% lower than those of the SPEA2 and NSGA-II algorithms and is lower by 30% using 32-thread and 128-thread parallel execution.

Conclusion

With the development of cloud computing, big data and artificial intelligence, cloud resource demands demonstrate the characteristics of diversity, burst and uncertainty. Undoubtedly, cloud platforms often encounter such emergent resource demands, which needs to be allocated resources quickly and optimally. This paper proposes a multi-objective optimization cloud resource allocation algorithm for emergent demands. The priority of resource allocation is first designed to respond to emergent demands, and resource performance and resource proportion matching distances are established to realize resource optimization and balanced utilization of all types of resources. Then, a multi-objective optimization algorithm of resource allocation is presented to guarantee the timeliness and optimization of resource allocation, in which a multi-objective mathematical model minimizes three objectives to optimize resource utilization and an improved NSGA-II algorithm accelerates the solving speed and improves the quality and distribution uniformity of the solution set. Experiments are performed to compare our proposed RAA-PI-NSGAII algorithm with the RR, BF, SPEA2 and NSGA-II algorithms. The results of this study verify the effectiveness of our algorithm to meet the emergent demands in cloud computing.

Acknowledgements
Not applicable.

Authors' contributions
Jing Chen is a major contributor in proposing the method, implementing simulation and drafting this manuscript. Tiantian Du carried out the partial experimental work. Gongyi Xiao contributed to the partial data analysis. The author(s) read and approved the final manuscript.

Funding
This work was supported by Project (ZR2020MF034, ZR2019QF014) supported by Shandong Provincial Natural Science Foundation.

Availability of data and materials
The datasets used during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 9 October 2020 Accepted: 9 February 2021
Published online: 01 March 2021

References

1. Barham P, Dragovic B, Fraser K et al (2003) Xen and the art of virtualization, ACM SIGOPS. Operating Syst Rev 37(5):164–177
2. Armbrust M, Fox A, Griffith R et al (2009) Above the clouds: a Berkeley view of cloud computing. University of California, EECS Department, University of California, Berkeley. In: UCB/EECS-2009-28
3. Pradhan P, Behera PK, Ray NNB (2016) Modified round Robin algorithm for resource allocation in cloud computing. Proc Comp Sci 85:878–890
4. Shirvastava S, Dubey R, Shrivastava M (2017) Best fit based VM allocation for cloud resource allocation. Int J Comp Appl 158(9):25–27
5. Katyal M, Mishra A (2014) Application of selective algorithm for effective resource provisioning in cloud computing environment. Int J Cloud Computing 4(1):1–10
6. Li Y, Tang X, Cai W (2016) Dynamic bin packing for on-demand cloud resource allocation. IEEE Trans Parallel Distributed Syst 27(1):157–170
7. Fatima A, Javaid N, Sultana T et al (2019) An efficient virtual machine placement via bin packing in cloud data centers. Adv Intell Syst Comput: 977–987
8. L. Guo, P. Du, A. Razaque et al., Energy saving and maximize utilization cloud resources allocation viaonline multi-dimensional vector bin packing, IEEE 2018 Fifth International Conference on Software Defined Systems (SDS) 2018, pp. 160–165
9. Vila S, Guirado F, Lerida JL et al (2019) Energy-saving scheduling on IaaS HPC cloud environments based on a multi-objective genetic algorithm. J Supercomput 75(3):1483–1495
10. Xu H, Liu Y, Wei W, Zhang W (2018) Incentive-aware virtual machine scheduling in cloud computing. J Supercomput 74(7):3016–3038
11. Nie J, Luo J, Yin L (2017) Energy-aware multi-dimensional resource allocation algorithm in cloud data center. KSII Trans Int Inform Syst 11(9): 4320–4333
12. Li Q, Hao Q, Xiao L, Li Z (2011) Adaptive management and multi-objective optimization for virtual machine placement in cloud computing. Chin J Comput 34(12):2253–2264
13. Zuo L, Shu L, Dong S, Chen Y, Yan L (2017) A multi-objective hybrid cloud resource scheduling method based on deadline and cost constrains. IEEE Access 99:1–13
14. Zhu W, Zhuang Y, Zhang L (2017) A three-dimensional virtual resource scheduling method for energy saving in cloud computing. Futur Gener Comput Syst 69:66–74
15. Xiao W, Bao W, Zhu X et al (2016) Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment. IEEE Trans Parallel Distributed Syst 27(7):1954–1967
16. Hinz M, Piegas KG, Miers CC et al (2018) A cost model for IaaS clouds based on virtual machine energy consumption. J Grid Comput 16(3):493–512
17. Zhu X, Yang LT, Chen H et al (2014) Real-time tasks oriented energy-aware scheduling in virtualized clouds. IEEE Transact Cloud Comput. 2(2):168–180
18. Dong J, Wang H, Li Y, Cheng S (2014) Virtual machine scheduling for improving energy efficiency in IaaS cloud. China Communications 11(3):1–12
19. Ye S, Wang T, Zhang W, Zhong H (2014) Profit-driven resource scheduling for virtualized cloud systems. IEEE/ACIS 13th International Conference on Computer and Information Science, pp 263–268
20. Zhao Y, Calheiros R, Gange G, Bailey J, Sinnott R (2015) SLA-based resource scheduling for big data analytics as a service in cloud computing environments, 2015 44th international conference on parallel processing (ICPP), pp 510–519
21. Bi J, Yuan H, Tan W et al (2015) Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center. IEEE Trans Autom Sci Eng 99:1–13
22. Zhu J, Li J, Zhuang Y (2015) Utility-based virtual cloud resource allocation model and algorithm in cloud computing. Int J Grid Distribut Comput 8(2): 177–190
23. Li D, Chen C, Guan J et al (2016) DCloud: deadline-aware resource allocation for cloud computing jobs. IEEE Transact Parall Distribut Syst 27(8): 2248–2260
24. Ali P, Mortez B, Saee SK (2019) An efficient method for allocating resources in a cloud computing environment with a load balancing approach. Concurrency Comput 31(17):e5285

25. Singh N, Dhindsa KS (2017) Hybrid scheduling algorithm for efficient load balancing in cloud computing. Int J Adv Networking Appl 8(5):3181–3187
26. Wei L, Huang T, Chen J, Liu Y (2013) Workload prediction-based algorithm for consolidation for virtual machines. J Electronics Inform Technol 35(6): 1271–1276
27. Tang F, Yang LT, Tang C, Li J, Guo M (2018) A dynamical and load-balanced flow scheduling approach for big data centers in clouds. IEEE Transact Cloud Comput 6(4):915–928
28. Shen D (2018) Research on application-aware resource management for heterogeneous big data workloads in cloud environment. Dongnan University
29. Hanani A, Rahmani AM, Sahafi A (2017) A multi-parameter scheduling method of dynamic workloads for big data calculation in cloud computing. J Supercomput 73(11):4796–4822
30. Gurleen K, Anju B (2018) A survey of prediction-based resource scheduling techniques for physics-based scientific applications. Modern Physics Letters B 32(25):1850295
31. Kaur G, Bala A (2019) An efficient resource prediction-based scheduling technique for scientific applications in cloud environment. Concurrent Eng Res Appl 27(2):112–125
32. Chen W, Lu YH, Hacker TJ (2015) Adaptive cloud resource allocation for analysing many video streams, IEEE 7th. Int Conf Cloud Comput Technol Sci:17–24
33. Laili YJ, Lin SS, Tang DY (2020) Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment. Robot Comput Integr Manuf 61: 101850
34. Li C, Tang J, Luo Y (2018) Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud. Clust Comput 21(2):1331–1348
35. Liu J, Ren J, Dai W et al (2019) Online multi-workflow scheduling under uncertain task execution time in IaaS clouds. IEEE Trans Cloud Comput 99:1–1
36. Reihaneh K, Faramarz SE, Naser N, Mehran M (2017) ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments. J Supercomput 73(6): 2430–2455
37. Chen H, Zhu J, Ma M, Zhu X (2017) Scheduling for stochastic tasks and resource in virtualized clouds. Syst Eng Electron 39(2):348–354
38. Chen H, Zhu J, Zhu X, Ma M, Zhang Z (2017) Resource-delay-aware scheduling for real-time tasks in clouds. J Comput Res Dev 54(2):446–456
39. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
40. Tan B, Ma H, Mei Y (2017) A NSGA-II-based approach for service resource allocation in cloud. IEEE Congress Evol Comput (CEC) 2017:2574–2581
41. Sofia AS, GaneshKumar P (2018) Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. J Netw Syst Manag 26(2):463–485
42. He Z, Dong J (2019) Virtual machine migration strategy for cloud data center based on NSGA-II, international conference on applications and techniques in cyber intelligence, pp 883–890
43. Xu X, Fu S, Yuan Y et al (2019) Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II. Comput Intell 35(3):476–495
44. Wen S, Zheng J (2010) Improved diversity maintenance strategy in NSGA-II. Comput Eng Appl 46(33):49–53

Publisher's Note