Journal of Communications and Information Networks
Vol.1, No.4, Dec. 2016
DOI: 10.11959/j.issn.2096-1081.2016.039

Research paper

Towards energy efficient cloud: an optimized ant colony model for virtual machine placement

ZHANG Liumei[1], WANG Yichuan[2], ZHU Lei[2], JI Wenjiang[2]

1. School of Computer Science, Xi'an Shiyou University, Xi'an 710065, China
2. Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

Abstract: A virtual machine placement optimization model based on optimized ant colony algorithm is proposed. The model is able to determine the physical machines suitable for hosting migrated virtual machines. Thus, it solves the problem of redundant power consumption resulting from idle resource waste of physical machines. First, based on the utilization parameters of the virtual machine, idle resources and energy consumption models are proposed. The models are dedicated to quantifying the features of virtual resource utilization and energy consumption of physical machines. Next, a multi-objective optimization strategy is derived for virtual machine placement in cloud environments. Finally, an optimal virtual machines placement scheme is determined based on feature metrics, multi-objective optimization, and the ant colony algorithm. Experimental results indicate that compared with the traditional genetic algorithms-based MGGA model, the convergence rate is increased by 16%, and the optimized highest average energy consumption is reduced by 18%. The model exhibits advantages in terms of algorithm efficiency and efficacy.

Key words: cloud computing, energy efficient, virtual machine placement, multi-objective optimization, ant colony optimization

Citation: ZHANG Liumei, WANG Yichuan, ZHU Lei, et al. Towards energy efficient cloud: an optimized ant colony model for virtual machine placement[J]. Journal of communications and information networks, 2016, 1(4): 116-132.

1 Introduction

Cloud computing improves the utilization rate of physical resources. It facilitates a single physical platform with rich system environments and application services via optimized integration and allocation of physical resources through virtualization technology. However, cloud service platforms are facing serious challenges such as excessive consumption of power, computing, storage, bandwidth, and other resources because the platform often provides all-weather real-time response services via centralized hardware devices, high-density application software, and massively complex computing tasks. According to the green and low-carbon initiative policies in China,

Manuscript received Aug. 17, 2016; accepted Nov. 25, 2016
This paper is supported by the National Natural Science Founds of China (No. 61602376), the Natural Science Research Project of Shaanxi Education Department (Nos. 16JK1573, 112-431016021), the Ph.D. Research Startup Funds of Xi'an University of Technology (Nos. 112-256081504, 112-451115002, 112-451116015), Research on the training mechanism of computer application ability of non computer majors in Petroleum Universities (No. SGH140627).

the primary goal of green cloud computing is to improve resource utilization (including networks, servers, storage, applications, and services), reduce the amount of physical equipment used, and reduce the energy consumption of traditional cloud platforms. Virtual machines are the core services provided by cloud computing platforms. Therefore, the virtual machine deployment strategy is the key to reducing the amount of physical equipment and improving physical resource utilization. Although scheduling and allocation of virtual resources are of utmost importance, the core idea of traditional the load balancing and scheduling strategy is to avoid overly heavy or light workloads on the physical server. It does not describe the resources utilization from the perspective of how much resources virtual machines take from the physical machine at a finite level. That results in the same physical machine often hosting virtual machines with both heavy and light resource utilization. If the virtual machines were to be classified according to utilization, and some selectively migrated, then some physical machines could be shut down, which would electively reduce the overall energy consumption of cloud platforms. Thus, on the premise of ensuring Service Level Agreement, this paper proposes a placement optimization model for cloud platforms that provides theoretical and technical support for the construction of low-power cloud computing platforms.

The remainder of this paper is organized as follows. In section 2, related research efforts are presented and summarized. Section 3 outlines how optimized virtual machine placement can be conducted in cloud computing. Section 4 explains the virtual machine optimization placement model and defines the optimization problem. Section 5 introduces our ant colony based multi-objective optimal algorithm for virtual machine placement. Section 6 presents the results of simulation experiments conducted to evaluate the feasibility and performance of the proposed algorithm. Section 7 concludes the paper.

2 Related work

Virtual machine placement is a process in which groups of virtual machines are mapped into a number of physical servers. Virtualization is one of the fundamental technologies of cloud computing. However, the problem of virtual machine placement has become a challenging issue in this area. Consequently, methods of optimizing the placement strategies have also became key research topics associated with improving resource utilization and energy efficiency. Many researchers have already underscored the importance of virtual machine placement in various studies[1,2]. The related methods proposed for solving virtual machine placement problems are categorized into four classes below.

Linear programming is widely used for traditional analysis. Based on linear and quadratic programming, Chaisiri et al. propose an optimized virtual machine placement method that reduces the number nodes used in cloud clusters[3]. In Refs.[4,5], linear programming is employed to describe the server consolidation problem. Constraint conditions have also been extended to allocation of virtual machines onto specific physical servers according to special attributes, thereby limiting the number of virtual machines accommodated on a physical server. This ensures restriction of the overall number of migrations and some virtual machines can be assigned to different physical servers. The LP-relaxation based heuristic method has also been applied to reduce the calculation overhead of application of linear programming.

Genetic algorithms have also been applied for virtual machine placement. GABA is an adaptive algorithm that automatically configures virtual machines in data centers[6]. It is able to effectively determine the optimal deployment location for virtual machines according to time varying demands and dynamic environmental conditions. In Ref.[7], the virtual machine placement is characterized as a

multi-objective optimization problem with the aim of reducing resource wastage, energy consumption, and heat dissipation cost. Based on fuzzy multi-objective evaluation, this method utilizes an improved genetic algorithm to effectively search a large solution space and integrates conflicting objectives conveniently.

Constraint programming has also been used for virtual machines placement under various environmental conditions. Van, et al. proposed a resource management framework based on the combination of utility based dynamic virtual machine manager and dynamic virtual machine placement manager[8]. The configuration and deployment of virtual machines are presented as two constraint satisfaction problems. An information entropy based resource manager has been proposed for homogeneous clusters. This resource manager enables dynamic integration via the constraint programming method. Further, it can resolve the problems of assigning the virtual machine to available nodes and how the virtual machines are migrating to these nodes.

Bin packing problem that usually describes the virtual machine deployment as an optimized vector packing problem. Many researchers have proposed heuristic methods to solve the problem[9-13]. For example, the pMapper system can pack virtual machines on a few physical machines to reduce the migration overhead. It solves the problem of energy consumption tradeoff under fixed performance constraints. The packing algorithm of pMapper is an extension of the FFD heuristic method. In addition, Feller et al. proposed a single-objective optimization algorithm based on MMAS. The proposed algorithm reduces the number of physical machines required without affecting services according to the current load.

Most of the proposed methods are appropriate for optimization of the placement of virtual machines in a single environment. However, placement in real environments needs to consider a number of factors, and multi objective virtual machine optimization and placement issues have become the main research issues in this field. Although hardware technology is developing, the energy consumption of cloud server still increases with the system load. The basic power support equipment of data centers also has a high energy consumption problem. Most of the energy consumption in data centers result from non-computing services. Fig.1 shows the relationship between server load and the energy consumption of three commonly used cloud servers. Fig.2 shows the average power consumption distribution of a server cluster in a typical cloud environment[14]. Among them, the CPU and memory power consumptions are 31% and 11%. Other power consumptions total 44%, which comprises heat dissipation, fan, and DC/DC and AC/DC power conversion electrical energy loss. In fact, when the processor utilization rate of the server is only 10%, it still consumes more than 50% of the peak power[15]. Similarly, when performance bottleneck of cloud clusters occurs, such as the disk, network, or any such resource, the idle power consumption of other resources will rise accordingly. Therefore, this paper proposes a virtual machine placement method for the randomly available virtual machine optimization placement problem as a multi-objective combinatorial optimization problem that simultaneously optimizes the over-all energy

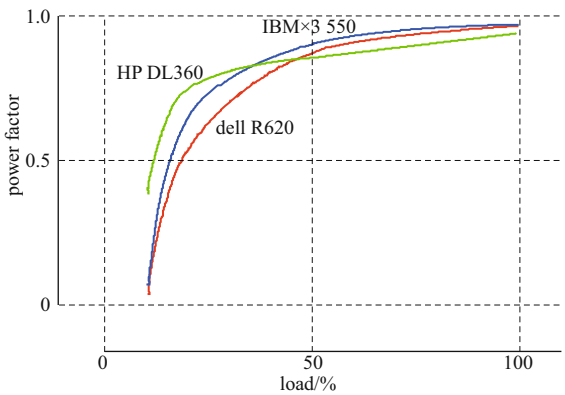

Figure 1 Relationship between system energy consumption and load

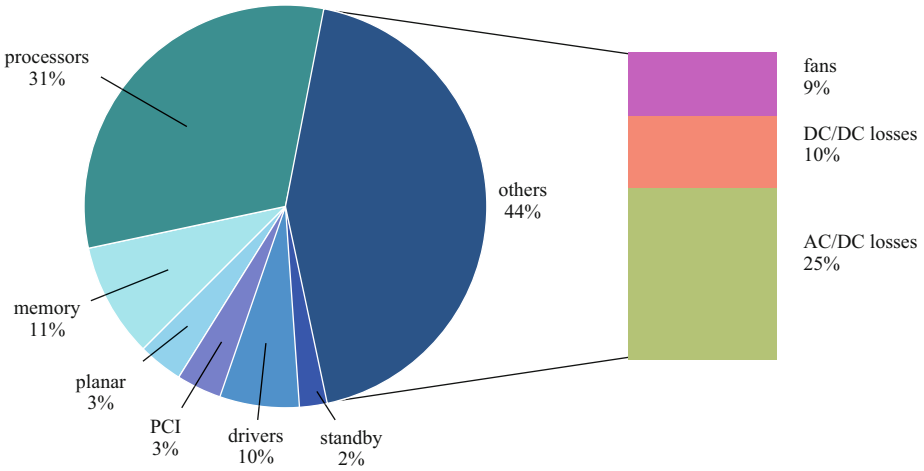

Figure 2 Average power consumption of the system

consumption and idle resources. An improved ant colony system-based algorithm is proposed to solve the potential problem of large scale solution space in large scale data centers.

3 Optimized virtual machine placement in cloud computing

Fig.3 demonstrates an optimization method for virtual machine placement in a typical virtualized environment. The cloud server cluster consists of four physical machines. Each server has physical resources to host virtual machine services. In the current scenario, the whole system consists of four virtual machines, each of which is used to run an independent application service. The calculation procedure for optimization of virtual machine placement is as follows[16].

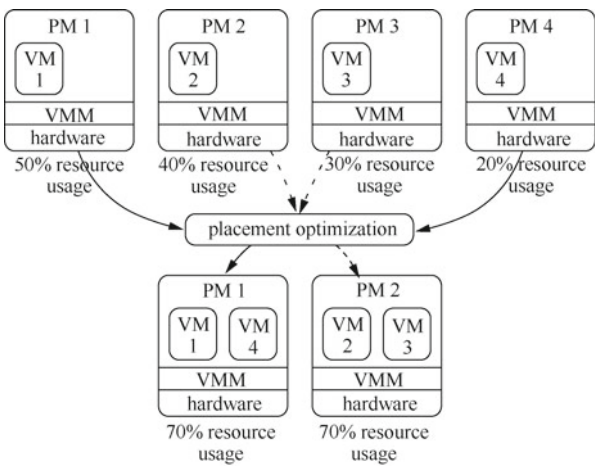

Figure 3 Optimized placement in virtualized environment

step 1 For each server, calculate the resource requirements of the application services using server resources utilization statistics.

step 2 Choose a target physical machine that has certain similarities to the source physical machine in terms of software, CPU type, network connectivity, and shared memory.

step 3 The candidate virtual machine will be deployed to the server in step 2. If the subsequent virtual machines are able to meet the resource requirement, it will also be deployed on the first server. If it does not satisfy the requirements, a different physical server will need to be introduced to accommodate more virtual machines, until all the virtual machines of the cluster have all been deployed.

step 4 The result of physical machine placement in step 3 is a server cluster under the specific working environment. Therefore, under the same service conditions, the number of physical machines decreases from four to two.

3.1 Ant colony optimization

ACO (Ant Colony Optimization) is a meta heuristic algorithm inspired by the collective foraging behavior of ants in a colony[17]. Ants are social insects whose behavior is influenced by the group's survival objectives, often between the nest and the food source. In the initial stage, the ants randomly explore the surrounding areas of the nest. While moving, they leave pheromone on the path to supply information for other ants. The concentration of pheromone is a key signal for the ants to find their way. When an ant finds a food source, it evaluates the quantity and quality of the food, and takes some back to the nest. On its return journey to the nest, the ant will leave an amount of pheromone corresponding to the quality and quantity of the food found. This pheromone guides other ants to the food source. Through this indirect communication between ants, ants can obtain the shortest path between the nest and the food source.

ACO has been applied to many study areas such as the traveling salesman problem[18], the flow shop scheduling problem[19] and the quadratic assignment problem[20]. In addition to the original combinatorial optimization field, ACO is also used to solve continuous optimization problems[21]. Extensions to ACO have also been proposed. Typical extensions include ACS[18], AS[22] and MMAS[23]. Researchers have also used ACO for multi objective optimization[24]. These algorithms are different in three aspects:

- Pheromone update. When updating the pheromone trail, it is necessary to select the pheromone release method from the construction solution. There are two kinds of pheromone trail updating strategies. The first strategy is to update the pheromone matrix with the iteration-best or best-so-far method for each objective. The second strategy is to collect and store the non-dominated solutions of the external set. Only the non-dominated solution method can update the pheromone.
- Definition of pheromone and heuristic information. In each solution construction step, the selection of candidate solutions usually depends on the probability between the pheromone and the heuristic factor. The definition of pheromone and the heuristic information is generally implemented using a single matrix or multiple matrices. When using a single matrix, each objective associated pheromone is integrated to convert the multiple objectives into a single objective. When multiple matrices are used, each matrix usually corresponds to an objective. For the pheromone information, each matrix can contain different values according to the implementation strategy used. This is also applicable to determination of heuristic information.
- Pheromone and heuristic information aggregation. When multiple matrices are being used, a variety of methods are required to aggregate the pheromone or the heuristic matrix. In general, there are three strategies: (1) weighted sum operation, in which aggregation is performed on the pheromone or the heuristic matrix, (2) weighted product operation, in which aggregation is performed on the pheromone or the heuristic matrix, and (3) in each step, the objective is optimized randomly. Whenever the weight value is used to aggregate multiple matrices, two strategies can be used to set the weight of the algorithm in each iteration: (a) dynamic weights, in which ,each ant can be assigned different weights by other ants, and (b) fixed weights, in which the weight of each ant is assigned equally and the weight of each objective is an equivalence at the runtime of the algorithm.

3.2 Multi-objective evolutionary algorithms

A MOEA(Multi-Objective Evolutionary Algorithm) is a stochastic optimization method that is often used to find optimal Pareto solutions. In the selection process, most of the existing multi-objective evolutionary algorithms are built on the dominant theory. Therefore, this paper focuses on the study of a multi-objective evolutionary algorithm based on dominance theory. The definition of dominant theory is as follows: In the case of no loss of generality, with m parameters (decision variables) and n objectives of the multi-objective minimization problem, the following description can be given:

$$\text{Minimze } \vec{f}(\vec{x}) = [f_1(x_1,\cdots,x_m),\cdots,f_n(x_1,\cdots,x_m)], \quad (1)$$

where

$$\vec{x} = (x_1,\cdots,x_m) \in X \vec{f} = (f_1,\cdots,f_n) \in Y. \quad (2)$$

Here $\vec{x}$ is the decision (parameter) vector in parameter space X, and $\vec{f}$ is the objective vector in objective space Y. At this point, the "solution" can be seen as a decision vector, and "points" as the corresponding objective vector. When the following conditions are true, it can be considered that $\vec{x}_1$ is the dominant solution of $\vec{x}_2$[25, 26].

- For any objective, solution $\vec{x}_1$ dominates $\vec{x}_2$.
- At least for one objective, solution $\vec{x}_1$ is more dominant than $\vec{x}_2$.

A point that is not dominated by any other point is called a non-dominated point. Such points usually form the optimal solution of the objective space. Thus, these points belong to the optimal solution space. Therefore, they can be called Pareto points, the corresponding vector can also be called the Pareto optimal solution.

The above concepts can be extended to search for non dominated solution sets. Assume there are N solutions, where each solution contains M(M>1) objective function values. The non dominated solutions set is obtained via the following steps[27]:

step 1 Let $i = 1$;

step 2 For $j \neq i$, for all objectives M, the solution of $\vec{x}_i$ and $\vec{x}_j$ is calculated by the above judgment condition, and the dominating solution is obtained;

step 3 For any j, if $\vec{x}_i$ is dominant to $\vec{x}_j$, then $\vec{x}_i$ is tagged, the iteration factor i is computed as i + +, then go to step 2;

step 4 Repeat the above operation, until the solution is complete, that is $i = N$, then go to step 5;

step 5 All the tagged solutions are non dominated solutions.

4 The virtual machine optimization placement model

A typical cloud environment consists of a server node pool and services running on it. Assuming that all the applications in the cluster are running on the virtual machine, then the virtual machine placement problem of the service node pool can be extended to the multi-dimensional vector packing problem. The dimension of the bin packing problem is the utilization of resources. As illustrated in Fig.2, the CPU power consumption and memory power consumption were 31% and 11%, with other power consumption at 44%, consisting of the cooling fan, DC/DC and AC/DC power conversion loss. (The study of other power consumption is beyond the scope of this paper.) Fig.1 demonstrates the relationship between energy consumption and load in three commonly used server cluster systems. When the system load increases, the overall energy consumption of the system also increases. This indicating that there is a linear relationship between load and power consumption.

Therefore, this paper describes the energy consumption of the virtual machine and the server node in two dimensions: CPU and memory attributes. The storage device is not considered because the general storage facility in the cloud environment mainly uses

storage servers as the main memory of clusters. If two virtual machines are running on the same physical machine, the CPU utilization rate of the server can be reacted by the sum of the CPU utilization of the two virtual machines. Similarly, the memory utilization rate of the physical machine can also be reacted by the sum of the memory utilization of the two virtual machines. For example, assume that a virtual machine's CPU utilization and memory occupancy rates are 10% and 20%, and the corresponding values for another virtual machine are 45% and 30%. Then, the corresponding attributes of the physical machine hosting the two virtual machines are 55% and 50%. That is the summary of the attribute vector of the virtual machine. In order to ensure that the CPU and memory utilization rates of the physical machine server are lower than the maximum load value of 100%, the threshold value of the physical ma- chine is set as the upper resource usage limit. Because, if the resource utilization rate of the physical machine was fully loaded, serious performance degradation would occur to affect the service level agreement. Meanwhile, the migration of virtual machines is also required to use CPU resources.

4.1 Resource idle model

The remaining available resources of each physical machine varies depending on the virtual machine deployed on it. In order to fully utilize these multi-dimensional resources, the following formula is used to calculate the potential cost of idle resources:

$$I_s = \frac{|R_s^p - R_s^m| + \rho}{C_s^p + C_s^m}, \tag{3}$$

where I_s is the resource idle rate of the physical machine s, O_s^p and O_s^m represent the normalized CPU and memory utilization respectively. R_s^p and R_s^m represent the normalized remaining CPU and memory resources respectively. ρ is a very small positive real number, with value set to 0.000 1. The core idea of the formula is to effectively utilize resources in all dimensions and balance the remaining resources on each server in different dimensions.

4.2 Energy consumption model

The power consumption of the server can be accurately described by the linear relationship between CPU utilization and power consumption[28]. In order to achieve energy saving, the server needs to be shut down when the server runs into an idle state. Therefore, the idle power consumption of the server is not an effective component of total energy consumption. Finally, the power consumption of the server s can be defined according to the CPU utilization function.

$$C_j = \begin{cases} (C_j^{\text{busy}} - C_j^{\text{idle}})(O_j^p + O_j^m) + C_j^{\text{idle}}, O_j^p, O_j^m > 0, \\ 0, \text{ otherwise}, \end{cases} \tag{4}$$

where C_j^{idle} and C_j^{busy} are the average idle consumption and average full load consumption of server j. O_j^p and O_j^m represent the normalized CPU and memory utilization respectively.

In addition, in a state of migration, the energy consumption varies when a virtual machine is migrated from physical server i to j.

$$\begin{cases} C_i = (C_i^{\text{busy}} - C_i^{\text{idle}})(O_i^p + O_i^m - \Delta O_i^p - \Delta O_i^m) + C_i^{\text{idle}}, \\ C_j = (C_j^{\text{busy}} - C_j^{\text{idle}})(O_j^p + O_j^m + \Delta O_j^p + \Delta O_j^m) + C_j^{\text{idle}}, \end{cases} \tag{5}$$

where, ΔO_i^p, ΔO_i^m is the energy needed when a virtual machine is migrated from physical machine i to j. $\Delta,O_j^p\Delta,O_j^m$ is roughly the same as $\Delta O_i^p\Delta,O_i^m$.

4.3 Definition of the optimization problem

In this section, optimization of virtual machine placement is formally described. Assume that

there are n virtual machines $i \in I$ to be placed on the m servers $j \in J$. For simplicity, it is assumed that no virtual machine requires computing resources that exceed the provided resource of a single physical machine. Further, assume that R_{pi} is the required CPU resource of each virtual machine, T_{pj} is the threshold of the CPU utilization rate of each physical server, R_{mi} is the required memory resource of each virtual machine, T_{mj} is the threshold of memory utilization rate of each physical server. Binary variable x_{ij} is adopted to indicate whether the virtual machine i is assigned to the physical machine j. y_j indicates the utilization of physical server j. The objective is then defined as reduction of the energy consumption and resource idle rate simultaneously. The formal definition of the optimization deployment problem is specified by Eqs.(6) and (7).

Eqs.(6) and (7) are subject to

$$\text{Minimize}\sum_{j=1}^{m} C_j = \sum_{j=1}^{m}\left[y_i \times \left((C_j^{\text{busy}} - C_j^{\text{idle}}) \sum_{j=1}^{n} (x_{ij} (R_{pi} + R_{mi})) + C_j^{\text{idle}} \right)\right], \tag{6}$$

$$\text{Minimize}\sum_{j=1}^{m} W_j = \sum_{j=1}^{m}\left[y_i \times \frac{\left| \left(T_{pj} - \sum_{j=1}^{m} (x_{ij} R_{pi}) \right) - \left(T_{mj} - \sum_{i=1}^{n} (x_{ij} R_{mi}) \right) \right| + \varepsilon}{\sum_{i=1}^{n} (x_{ij} R_{pi}) + \sum_{i=1}^{n} (x_{ij} R_{mi})} \right], \tag{7}$$

$$\text{Minimize}\sum_{j=1}^{m} C_j \approx \sum_{j=1}^{m} y_i C_j^{\text{idle}} + \sum_{j=1}^{m} (C_j^{\text{busy}} - C_j^{\text{idle}}) \sum_{j=1}^{m} x_{ij} (R_{pi} + R_{mi}), \tag{8}$$

$$I \bigcup J = \varnothing, \tag{9}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i \in I, \tag{10}$$

$$\sum_{j=1}^{m} R_{pi} x_{ij} \leqslant T_{pj} y_j, \forall j \in J, \tag{11}$$

$$\sum_{j=1}^{m} R_{mi} x_{ij} \leqslant T_{mj} y_j, \forall j \in J, \tag{12}$$

$$y_j, x_{ij} \in \{0,1\}, \forall j \in J \text{ and } \forall i \in I, \tag{13}$$

where, constraint in Eq.(9) indicates that the virtual machine moved out will no longer move back to the original physical machine. This is defined as the continual work of our studies[29] for reducing the overall energy consumption of the cloud platform. Constraint in Eq.(10) limits virtual machine i to only be assigned to one physical machine. Constraint in Eqs.(11) and (12) depict the capacity condition of the physical machine. Constraint in Eq.(13) define the variable domain of the problem. When given virtual machine set n and the physical machine set m, in fact, there are m^n schemes for optimizing virtual machine deployment.

For Eq.(4), there are m^n optimization schemes. From Fig.1 it is clear that there is little difference between the measurement factors of energy consumption of the physical machines when the load reaches a certain threshold. For real case analysis, the energy consumption of the physical machine moving out of the virtual machine set and the energy consumption of the other physical machine after moving into the virtual machine can be regarded as an approximate value. That is $\Delta O_i^p \approx \delta O_j^p$ and $\Delta O_i^m \approx \delta O_j^m$, therefore $C_i^{\text{busy}} \approx C_j^{\text{idle}}$ and $C_i^{\text{idle}} \approx C_i^{\text{busy}}$. Then according to Eq.(5), migration will appear as follows:

$$\begin{cases} C_i = (C_i^{\text{busy}} - C_i^{\text{idle}}) (O_i^p + O_i^m - \Delta O_i^p - \Delta O_i^m) + C_i^{\text{idle}}, \\ C_j = (C_i^{\text{busy}} - C_i^{\text{idle}}) (O_i^p + O_i^m + \Delta O_i^p + \Delta O_i^m) + C_i^{\text{idle}}. \end{cases} \tag{14}$$

When $O_i^p + O_i^m - \Delta O_i^p - \Delta O_i^m = 0$, which indicates that all of the virtual machines on a certain physical machine have been migrated, the physical machine can be powered off. However, the actual situation is that the physical machine's idle energy consumption accounts for 70% of the average energy consumption of the full load. This is also confirmed in the literature[30]. In order to reduce the overall objective numbers, the

objective function can be minimized; thus, Eq.(6) is optimized as Eq.(8).

In fact, when all the physical machines have the same configuration, the above equation can be satisfied. Then, the number of y_i will be reduced to one, to achieve objectives minimization, according to Eq.(8), the literature[30], and ACO. Then the method proposed in Ref.[30] can be used for the migration to achieve the goal of physical machine shutdown.

5 Ant colony based multi-objective optimal algorithm for virtual machine placement

A complete and feasible solution to the optimization problem of the multi objective virtual machine placement is to obtain the effective virtual machine arrangement scheme. Therefore, based on ACO, this paper proposes an optimal multi-objective method, ACVMP, for virtual machine placement. Algorithm 1 describes the pseudo code as follows: During the initialization of the algorithm, all the parameters are initialized, and the pheromone trail is set to τ_0. In the iteration of the algorithm, each ant receives the request of all the virtual machines, and then calls a physical server to distribute the virtual machine. This is achieved by using the pseudo random proportional rule to describe the tendency of the ants to place a particular virtual machine on the host. This rule is based on the heuristic basis of pheromone concentration in the current path and on the basis of guiding the ants choosing the best heuristic advantage. When an ant creates a path, the corresponding local pheromone is updated. When all the ants have completed construction of their solutions, each solution to the current Pareto set is updated. Assignment of a virtual machine to a physical machine is called a movement, and is represented by $VM \gtrdot PS$.

5.1 Definition of pheromone and heuristic information

As with the ACO algorithm, the ACVMP algorithm takes the pheromone matrix and the heuristic information matrix as the initial step. The quality of the ACO algorithm depends largely on the definition of pheromone trail[22]. The fundamental problem is the definition of the problem characteristics. Two possible pheromone structures exist: 1) the pheromone trail is associated with $VM \gtrdot PS$ path; 2) the pheromone trail is associated with each virtual machine pair. In this paper, we only use the first possibility to release pheromone, that is to define the pheromone trail $\tau_{i,j}$ by referencing the benefits of migrating virtual machine i to physical machine j. The initial pheromone concentration can be obtained from $\tau_0 = \frac{1}{[n \cdot (C'(S_0) + I(S_0))]}$, where n is the number of virtual machines, S_0 is the solution generated by the FFD heuristic method, $I(S_0)$ are the resources idle rate towards solution S_0, $C'(S_0)$ is the energy consumption of solution S_0 after the normalization. The calculation is as follows:

$$C'(S_0) = \sum_{j=1}^{m} (C_j / C_j^{\text{Max}}), \tag{15}$$

where, C_j^{Max} is the peak energy consumption for server j.

In addition to pheromone, another important consideration in the use of the ACO is to select the appropriate heuristic method. The heuristic method is combined with the pheromone information to obtain a solution. It will become an important basis for the ant to construct the probability solution. In this paper, $\eta_{i,j}$ is adopted to describe the heuristic information and characterize the desirability of allocating virtual machine i to host j. In order to accurately evaluate the desirability of the movement of ants, the current state of the ants is calculated dynamically to obtain heuristic information. As the calculation of heuristic

information needs to traverse all the movements of the ants, it significantly affects the efficiency of the algorithm. In order to avoid this, an algorithm is needed to calculate the heuristic information effectively. Therefore, a heuristic information algorithm that fully considers the contribution of every movement of ants to the value of the overall objective function is proposed. Let *PMC* be the cluster of all physical machines. When constructing the solution, each ant will have all the virtual machine set and random *PMC* set. The ant will assign the virtual machine to the first physical machine in *PMC* individually. When the first physical machine is not able to accept any more virtual machines, the ant will assign the remaining virtual machines to the second physical machine until all virtual machines are assigned. Therefore, when calculating $\eta_{i,j}$, the placement of virtual machines from host 1 to host *j* has a known order. Then, the local contribution value of the first objective function for assignment of virtual machine *i* to physical machine *j* can be calculated by the following equation:

$$\eta_{i,j,1} = \frac{1}{\varepsilon + \sum_{v=1}^{j} (C_v / C_v^{\text{Max}})}. \tag{16}$$

Accordingly, the local contribution value of the second objective function for assignment of virtual machine *i* to physical machine *j* can be calculated by the following equation:

$$\eta_{i,j,2} = \frac{1}{\varepsilon + \sum_{v=1}^{j} W_v}. \tag{17}$$

Algorithm 1 Part 1 ACVMP algorithm

Require:

Virtual machine collection;

Physical machine collection (including details of resource requirements and configuration);

Resource utilization threshold;

Ensure:

Pareto solution set *P*;

/*Algorithm initialization*/

1: Initialize parameters ρ_1, ρ_2, α, τ_0, q_0, Number of ants *N*, Number of Iterations *M*

2: Empty *P*

3: Set all pheromone values to τ_0

/*Iterations*/

4: **repeat**

5: **for** $j = 1$ to *N* **do**

6: Random sort physical machine list *PMC*

/*solution construction*/

7: **repeat**

8: Select a new physical machine from set *PMC*

9: **repeat**

10: **for each** Virtual machine that can be deployed to the current physical machine **do**

11: Based on in Eq.(18) calculate the overall desirability of the virtual machine to be deployed on the physical machine

12: Based on in Eq.(21) calculate the probability of the ants to deploy the virtual machine on the physical machine

13: **end for**

/*Determine the deployment of virtual machines*/

14: Draw *q*

Algorithm 1 Part 2 ACVMP algorithm

15: **if** $q \leqslant q_0$ then

16: exploitation

17: **else**

18: exploration

19: **end if**

/*Updated local pheromone*/

20: Based on in Eq.(22) apply local update rule

21: **until** The current physical machine is not suitable for the deployment of any remaining virtual machine

22: **until** All virtual machines are deployed

23: **end for** /*Evaluation process*/

24: Calculate the two target values of each solution under the current ant colony size

25: If the current ant colony size is not dominated by any other solution or Pareto solution, the current solution is added to set P, then remove the remaining dominating solutions from set P

/*Global pheromone update*/

26: **for each** Non dominated solution of Pareto's solution set P **do**

27: Based on in Eq.(23) apply global update rule

28: **end for**

29: **until** reach the set value

30: **return** Pareto solution set P

For the multi-objective optimization problem, the overall desirability of each movement is generally obtained through the integrated local desirability, and there are many ways to integrate it. In this paper, an integration method to describe the overall desirability of allocating virtual machine i to physical machine j is proposed:

$$\eta_{i,j}=\eta_{i,j,1}+\eta_{i,j,2}. \tag{18}$$

5.2 Problem solution

For a collection of virtual machines that needs to be placed, the ant will select virtual machine i as the target for the next load of current host machine j according to the following pseudo random proportion rule.

$$j=\begin{cases}\arg\max_{u\in\Omega_k(j)}\alpha\tau_{u,j}+(1-\alpha)\tau_{u,j}, q\leqslant q_0,\\ s, \text{otherwise}.\end{cases} \tag{19}$$

where, α is a parameter that allows the user to control the relative importance of the pheromone trail, q is a random number evenly distributed in the range [0,1]. When $q > q_0$ the process is called exploration; conversely, when $q<q_0$ the process is called exploitation. q_0 determines the fixed parameters by means of the relative importance of the exploration and exploration of the accumulated knowledge of the problem ($0 \leqslant q_0 \leqslant 1$). $\Omega_k(J)$ is a virtual machine set that meets the conditions of the deployment of host machine j. Thus,

$$\begin{aligned}&\Omega_k(j)=i\in 1,\cdots,n\,|\,(\sum_{u=1}^{m}x_{iu}=0)\wedge\\&((\sum_{u=1}^{n}(x_{uj}R_{pu})+R_{pi}\leqslant T_{pj})\wedge\\&((\sum_{u=1}^{n}(x_{uj}R_{mu})+R_{mi}\leqslant T_{mj}),\end{aligned} \tag{20}$$

$\eta_{i,j}$ was defined in Eq.(18). The pheromone value $\tau_{i,j}$ is given in Eq.(23), where s is a random variable that is selected according to random probability[31]. In fact, it is the probability of ant k selecting virtual machine i to place on host machine j:

$$p_{i,j}^{k}=\begin{cases}\dfrac{\alpha\tau_{i,j}+(1-\alpha)\eta_{n,j}}{\max\limits_{u\in\Omega_k(j)}(\alpha\tau_{u,j}+(1+\alpha)\eta_{u,j})}, i\in\Omega_k(j),\\ 0, \text{otherwise}.\end{cases} \tag{21}$$

There are two reasons for the calculation of the probability of selection by the method mentioned above. First, in order to simplify the process, Ref.[31] proposes the relative importance of using a single control parameter (α) to map the pheromone quality and the desirability of each path. Secondly, the computation efficiency of the proposed method is improved by using multiplication operation instead of the exponential operation.

5.3 Pheromone update

Another core importance of the ACVMP algorithm is the pheromone trail update. Pheromone trail value increases with the pheromone released from the ants, and it decreases with pheromone evaporation. The release of new pheromone is generally based on the fact that some of the advantage solution contain information that needs to be tracked through pheromone trails. The path information of the advantage solution can be deviated as other ants construct subsequent solutions. However, pheromone

evaporation also contains a practical forgetting function, which avoids the fast convergence of the algorithm in the suboptimal region, which is favorable for the exploration of new fields in the search space. This is a kind of diversification strategy. The algorithm proposed in this paper includes two steps: local and global pheromone update. When performing the assignment of virtual machine i to host machine j, ants will reduce the pheromone trail concentration between the path of virtual machine i and host machine j with the following local update rules:

$$\tau_{i,j}(t) = (1-\rho_l)\tau_{i,j}(t-1) + \rho_l \tau_0 + \frac{1}{(I_j+1)(\log m+1)}, \quad (22)$$

where, τ_0 is the initial pheromone concentration, $\rho_l (0<\rho_l<1)$ is the parameter of local pheromone evaporation, I_j is the idle rate of physical machine j, m is the total number of physical machines.

Global update rules are applied when all ants have completed construction of the solution. Because all non dominated solutions or Pareto solutions can be considered to be the optimal solution for the multi-objective optimization problem, this paper assumes that all the non dominated solutions have the same high quality and ignore all the dominated solutions. Therefore, for each solution of the current Pareto solution S, the global update method is calculated by the following equation:

$$\tau_{i,j}(t) = (1-\rho_g)\tau_{i,j}(t-1) + \frac{\rho_g \lambda}{P'(S)+W(S)}, \quad (23)$$

subject to,

$$\lambda = \frac{N}{t - NI_s + 1}, \quad (24)$$

where, $\rho_g (0<\rho<1)$ is the global update parameter of pheromone evaporation. The global non dominated solutions of the Pareto set will be stored in the external set. If the solution in the current iteration is not dominated by any other solution or other non dominated solutions in the external set, the solution is added to the external set and increases the number of pheromones that contains the solution path. Then, in an external set, all other solutions that are dominated by this solution will be cleared. In Eq.(24), N is the number of ants, NI_S represents the solution, and S has joined the external collection. λ is a self-adaptive coefficient, that is used to control the contribution of the solution to the pheromone information in the external set. This global updating rule can increase the learning ability of the ants.

6 Experimental results and analysis

In this section, the feasibility and performance of the proposed algorithm are evaluated via simulation experiments. For performance comparison, the proposed ACVMP algorithm and two other algorithms, MGGA[32] and VMPACS[30], were implemented in Python. The algorithms were executed on a Lenovo M8400T Think Center computer with 3.39 Intel Core i7GHz processor, 8 GB RAM, and operating system Fedora 22, as shown in Tab.1.

The performance of the algorithm is directly affected by the setting of various parameters of ACVMP. In this study, the appropriate parameter values were determined by preliminary experiments: $N = 10$, $M = 100$, $\alpha = 0.45$, $\rho_l = \rho_g = 0.35$ and $q_0 = 0{:}8$. Referring to VMPACS, the population size was set to 12, the initial population was generated by a random method, the crossover rate was 0.7, the mutation rate was 0.005, the maximum number of generations for each search procedure was 10. In contrast to VMPACS, the threshold of the CPU and memory utilization were set to $T_{pj} = T_{mj} = 75\%$ in the entire experiment. This is because the minimum lower bound for the optimal operating state of the processor is 70%[33]. Therefore, the threshold value 90% used in VMPACS, should be carefully considered. Because,

Table 1 Experiment configuration

	model	number	CPU	memory	OS
control node	Lenovo ThinkCenter M8400T	1	Intel core I7 3.39 GHZ	8 GB, DDR3 1 600 Mhz	fedora 22 workstation
physical server	Lenovo ThinkServer RD550	10	Xeon E5-2609 2 GHz	16 GB, DDR4 2 133 Mhz	CentOS 7
virtual machine		20	kvm vcpu	512 MB	fedora 22 server

there should be a serious impact on the physical machine after the migration when the CPU runs at high load.

Tab.1 shows the hardware configuration of the experimental environment. The cloud platform used was OpenStack, with 10 physical machines, all Think Server RD550 models with operating system CentOS 7. Each physical machine server hosted two virtual machines. The virtualization environment was KVM. The virtual machine operating system was Fedora 22 server. All virtual machines ran the same HTTP Apache service configuration. Using the Pylot load testing tool, test plans were set respectively for different virtual machines. To simulate the actual load conditions, all virtual machines had different memory and CPU resource requirements. For simplicity, the experiment was arranged in a homogeneous physical server environment (this does not mean that the proposed method can not be used in a heterogeneous server environment). When there are multiple non-dominated solutions after the operation of the algorithms, the solution is randomly selected from the non-dominated solution set.

In the initialization process, the experimental data were measured using a the power analyzer on idle energy consumption of 10 physical machine servers. Then, the HTTP services of 20 virtual machines were called through the Pylot. At this point, the status of the virtual machine servers can be seen as the normal operation of the entire cloud environment. Then, the power analyzer readings, and the CPU and memory utilization parameters of each physical machine server were recorded. The experimental dataset thus can be imported into the algorithm operating environment. Successively, the Python language was used to call the OpenStack API according to the results of the algorithm for live migration of the virtual machines. Then, physical machines that had no virtual machines were shut down, and the power analyzer readings recorded.

Experiments were repeated for four times using ACVMP and two other algorithms. Each runs was two minutes. Pylot were set different parameters for load call. Fig.4 is the experimental result for the ACVMP algorithm. Fig.4(a) is the data sampling plot of the first 60 seconds of the power analyzer. The x axis is the time unit in seconds, the y axis is the sum of the power analyzer attached to the power interface of each physical machine. It can be seen that the total energy consumption of the physical machine cluster achieves a gradient descent in about 30 seconds after starting the experiment. The first point at which energy consumption starts to decrease is about 8 seconds, and it stops at about 15 seconds. This is because after the start of the algorithm, it is necessary to calculate the current iteration of the contents in the optimal solution. Then, the virtual machine is deployed; the virtual machines that needs to be live migrated should be selected, the original physical machine should be powered off. The live migration generally took 2~4 seconds, and the power off generally took 3~5 seconds. This is because the virtual machine used the Fedora 22 server system which does not contain a graphical user interface, and

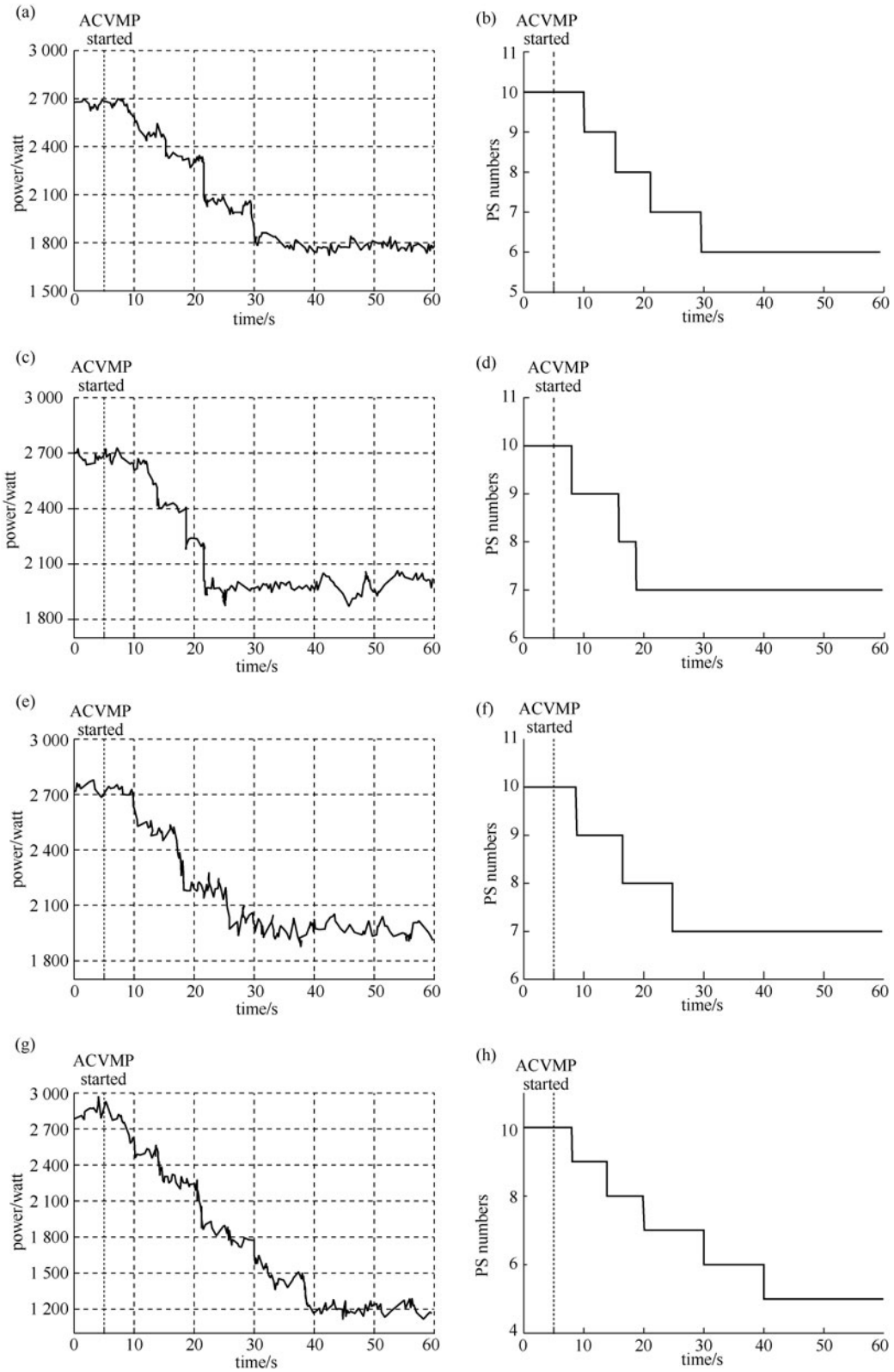

Figure 4 Results of optimization of VM placement: (a) power meter of test batch 1; (b) physical server numbers of test batch 1; (c) power meter of test batch 2; (d) physical server numbers of test batch 2; (e) power meter of test batch 3; (f) physical server numbers of test batch 3; (g) power meter of test batch 4; (h) physical server numbers of test batch 4

virtual machine storage is configured on an additional NAS server. After 30 seconds, the overall energy consumption became smooth, which shows that the algorithm is complete, and the optimal number of servers to power off has been identified. Figure4(a) shows an obvious effect that, in the case of ensuring the availability and stability of cluster services, the overall energy consumption decreased by about 900 watts. Fig.4(b) shows the number of virtual machines varying in the first 60 seconds. The x axis is time, and the y axis is the number of physical machines. Through the comparison in Fig.4, it is obvious that the number of physical machines has a significant impact on the overall energy consumption of the cluster. In comparison with he VMPAC and MGGA algorithms for four runs of the experiment, the ACVMP algorithm proposed in this paper has a relatively high shutdown ratio and the best effect, as can be seen in Tab.2.

Table 2 Comparison of physical machine numbers

	batch 1	batch 2	batch 3	batch 4	shutdown rate
ACVMP	6	7	7	5	0.375
VMPAC	7	8	7	6	0.3
MGGA	8	8	7	8	0.225

In terms of energy consumption, in order to compare the application effect of the three algorithms, the average value of the four experimental results was computed, and the average energy of 60 seconds after the start of the experiment was plotted, as shown in Fig.5. For algorithm cost, we configured the following scenarios to compare them in terms of time, which is represented by convergence rate. The three algorithms were set the same starting time. It can be seen that about 2~4 seconds after application of the algorithms, the overall energy consumption of the system begins to decrease. VMPAC obtains the optimized solution slightly earlier than the proposed ACVMP. This is because VMPAC obtained fewer physical machines for power o than ACVMP. Among the three algorithms, MGGA has the slowest convergence rate, which is as a result of the nature of the genetic algorithm. In roughly the same case, compared with VMPAC, the ACVMP algorithm proposed in this paper further reduces the energy consumption by about 5%, and improves the convergence speed by about 0.05%. Compared with MGGA, ACVMP can further reduce the energy consumption by about 18%, and improves the convergence speed by about 16%. In this scenario, the application results and performance of this algorithm are higher than those of the other two algorithms, and is closer to the optimal solution.

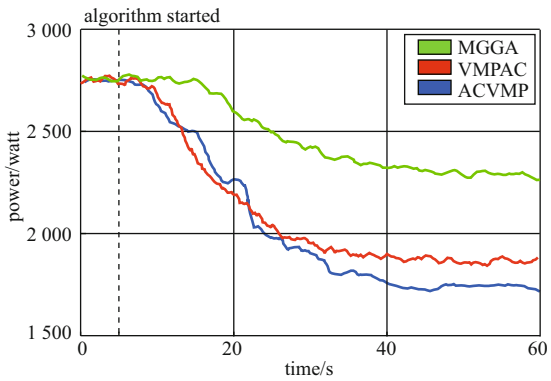

Figure 5 Effect comparison among algorithms

7 Conclusion

In this paper, a virtual machine placement optimization problem was studied in the context of green cloud computing. Consequently, an ant colony optimization model was proposed for virtual machine deployment. The model can be used to map virtual machines to suitable physical machines considering resource utilization. The model extends the optimization placement problem to a multi-objective optimization problem with multidimensional vector packing. By means of two sub-models, resource idle and energy consumption, the model can be utilized to find an optimal placement scheme. Experiments conducted

via simulations verify the effectiveness of the model and algorithm. Compared with similar models, the convergence rate was shown to have improved by a maximum of 16% , and average energy consumption reduced by as much as 18%. The experimental results also show that the algorithm proposed in this paper can indeed play a useful green, energy-saving role.

References

[1] CARDOSA M, KORUPOLU M R, SINGH A. Shares and utilities based power consolidation in virtualized server environments[C]// IFIP/IEEE International Symposium on Integrated Network Management, 2009: 327-334.

[2] GRIT L, IRWIN D, YUMEREFENDI A, et al. Virtual machine hosting for networked clusters: building the foundations for autonomic orchestration[C]//The 2nd International Workshop on Virtualization Technology in Distributed Computing, 2006: 7.

[3] CHAISIRI S, LEE B S, NIYATO D. Optimal virtual machine placement across multiple cloud providers[C]//IEEE Asia-Pacific Services Computing Conference, 2009: 103-110.

[4] BICHLER M, SETZER T, SPEITKAMP B. Capacity planning for virtualized servers[C]//Workshop on Information Technologies and Systems (WITS), 2006.

[5] SPEITKAMP B, BICHLER M. A mathematical programming approach for server consolidation problems in virtualized data centers[J]. IEEE transactions on services computing, 2010, 3(4): 266-278.

[6] MI H,WANG H, YIN G, et al. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers[C]// IEEE International Conference on Services Computing (SCC), 2010: 514-521.

[7] XU J, FORTES J A. Multi-objective virtual machine placement in virtualized data center environments[C]//IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, 2010: 179-188.

[8] VAN H N, TRAN F D, MENAUD J M. Performance and power management for cloud infrastructures[C]//IEEE 3rd International Conference on Cloud Computing (CLOUD), 2010: 329-336.

[9] BOBROFF N, KOCHUT A, BEATY K. Dynamic placement of virtual machines for managing sla violations[C]//The10th IFIP/IEEE International Symposium on Integrated Network Management, 2007: 119-128.

[10] VERMA A, AHUJA P, NEOGI A. pMapper: power and migration cost aware application placement in virtualized systems[C]//ACM/ IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, 2008: 243-264.

[11] SRIKANTAIAH S, KANSAL A, ZHAO F. Energy aware consolidation for cloud computing[C]//Conference on Power Aware Computing and Systems, 2008: 10.

[12] LI B, LI J, HUAI J, et al. Enacloud: an energy-saving application live placement approach for cloud computing environments[C]// IEEE International Conference on Cloud Computing, 2009: 17-24.

[13] FELLER E, RILLING L, MORIN C. Energy-aware ant colony based workload placement in clouds[C]//IEEE/ACM 12th International Conference on Grid Computing, 2011: 26-33.

[14] INTEL. Power management in Intel R architecture servers[EB/OL]. http://download.intel.com/support/motherboards/server/sb/power management of intel architecture servers.pdf, 2009

[15] CHEN G, HE W, LIU J, et al. Energy-aware server provisioning and load dispatching for connection-intensive internet services. [C]//The 5th Usenix Symposium on Networked Systems Design & Implementation, 2008: 337-350.

[16] KHANNA G, BEATY K, KAR G, et al. Application performance management in virtualized server environments[C]//The 10th IEEE/ IFIP Network Operations and Management Symposium, 2006: 373-381.

[17] LIN C, WU G, XIA F, et al. Energy efficient ant colony algorithms for data aggregation in wireless sensor networks[J]. Journal of computer and system sciences, 2012, 78(6): 1686-1702.

[18] DORIGO M, GAMBARDELLA L M. Ant colony system: a cooperative learning approach to the traveling salesman problem[J]. IEEE transactions on evolutionary computation, 1997, 1(1): 53-66.

[19] SHYU S J, LIN B M, YIN P Y. Application of ant colony optimization for nowait owshop scheduling problem to minimize the total completion time[J]. Computers & industrial engineering, 2004, 47(2): 181-193.

[20] MANIEZZO V, COLORNI A. The ant system applied to the quadratic assignment problem[J]. IEEE transactions on knowledge and data engineering, 1999, 11(5): 769-778.

[21] SOCHA K, BLUM C. An ant colony optimization algorithm for continuous optimization: application to feed forward neural network training[J]. Neural computing and applications, 2007, 16(3): 235-247.

[22] DORIGO M, MANIEZZO V, COLORNI A. Ant system: optimization by a colony of cooperating agents[J]. IEEE transactions on systems man & cybernetics part B cybernetics a publication of the IEEE systems man & cybernetics society, 1996, 26(1): 29-41.

[23] STÜUTZLE T, HOOS H. Max-min ant system and local search for the traveling salesman problem[C]//IEEE International Conference on Evolutionary Computation, 1997: 309-314.

[24] GARCIA-MARTINEZ C, CORD_ON O, HERRERA F. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp[J]. European journal of operational research, 2007, 180(1): 116-148.

[25] BRANKE J, DEB K, MIETTINEN K, et al. Multi-objective optimization: interactive and evolutionary approaches[M]. Berlin: Springer Berlin Heidelberg, 2008.

[26] DEB K. Multi-objective optimization using evolutionary algorithms: volume 16[M]. Hoboken: John Wiley & Sons, 2001.

[27] DEB K. Multi-objective genetic algorithms: problem difficulties and construction of test problems[J]. Evolutionary computation, 1999, 7(3): 205-230.
[28] FAN X, WEBER W D, BARROSO L A. Power provisioning for a warehouse-sized computer[C]//ACM SIGARCH Computer Architecture News, 2007: 13-23.
[29] ZHANG L M, MA J F, WANG Y C, et al. Toward green cloud computing: an attribute clustering based collaborative filtering method for virtual machine migration[J]. Information technology journal. 2013, 12(23): 7275.
[30] GAO Y, GUAN H, QI Z, et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing[J]. Journal of computer and system sciences, 2013, 79(8): 1230-1242.
[31] MANIEZZO V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem[J]. INFORMS journal on computing, 1999, 11(4) :358-369.
[32] XU J, FORTES J A B. Multi objective virtual machine placement in virtualized datacenter environments[C]//The 2010 IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing, 2010: 179-188.
[33] LIU C L, LAYLAND J W. Scheduling algorithms for multiprogramming in a hard real-time environment[J]. Readings in hardware/software Co-design, 2002, 20(1): 179-194.

About the authors

ZHANG Liumei [corresponding author] was born in Weinan, China. He received a B.E. degree in computer science from Air Force Engineering University; an M.I.T degree in information technology from the University of Sydney, and a Ph.D. degree from Xidian University. He is currently a lecturer in Xi'an Shiyou University. His research interests include data-mining and cloud computing. (Email: zhangliumei@xsyu.edu.cn)

WANG Yichuan was born in Kaifeng, China. He received a Ph.D. degree in computer system architecture from Xidian University of China in 2014. He is currently a Lecturer in Xi'an University of Technology and is also with Shaanxi Key Laboratory of Network Computing and Security Technology. His research interests include cloud computing, trusted computing, and networks security. (Email: chuan@xaut.edu.cn)

JI Wenjiang was born in Yanan, China. He obtained B.S. and Ph.D. degrees from Xidian University in 2006 and 2013, respectively. He is currently a lecturer in Xi'an University of Technology. His research interests include information and network security in VANET, privacy preservation in VANETs and network simulation. (Email: wjj@xaut.edu.cn)