Hindawi
Scientific Programming
Volume 2018, Article ID 9471356, 10 pages
https://doi.org/10.1155/2018/9471356

Research Article

Energy-Aware VM Initial Placement Strategy Based on BPSO in Cloud Computing

Xiong Fu[1], **Qing Zhao**[1], **Junchang Wang**[1], **Lin Zhang**[1] and **Lei Qiao**[2]

[1]*School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*Beijing Institute of Control Engineering, Beijing 100190, China*

Correspondence should be addressed to Xiong Fu; fux@njupt.edu.cn

Received 22 September 2017; Revised 21 December 2017; Accepted 10 January 2018; Published 14 February 2018

Academic Editor: Emiliano Tramontana

In recent years, high energy consumption has gradually become a prominent problem in a data center. With the advent of cloud computing, computing and storage resources are bringing greater challenges to energy consumption. Virtual machine (VM) initial placement plays an important role in affecting the size of energy consumption. In this paper, we use binary particle swarm optimization (BPSO) algorithm to design a VM placement strategy for low energy consumption measured by proposed energy efficiency fitness, and this strategy needs multiple iterations and updates for VM placement. Finally, the strategy proposed in this paper is compared with other four strategies through simulation experiments. The results show that our strategy for VM placement has better performance in reducing energy consumption than the other four strategies, and it can use less active hosts than others.

1. Introduction

As a new commercial calculation model, cloud computing is the evolution of distributed computing, parallel computing, and grid computing. With the advent of the era of big data, data has become a strategic resource of information society, and cloud computing can be more economical, effective, and efficient for mining data value, which has a revolutionary impact on human economic and social development. However, problems of high energy consumption have been given more and more attentions by enterprises and experts when the scale of data centers is constantly expanding. The increasing scale of data center mainly causes two kinds of serious energy problems. On the one hand, increasing physical servers can bring about more energy consumption; on the other hand, if a server loads few virtual machines or has few demands for resources, it will result in low utilization of server's resources, which will cause a huge waste of electric power. At present, rapid development of virtualization technology provides a new solution for power consumption in a data center. In particular, when cloud computing becomes a main developing direction in the future, due to its own server consolidation, online migration, isolation, high availability, flexible deployment, low administrative overhead, and other advantages, there is more space for the development of virtualization. Cloud computing uses mature virtualization technology, which makes resources it uses required in the form of virtual machines and then allocates servers' part resources to them to perform corresponding tasks, so resource-scheduling process can be converted into the search process of virtual machines [1]. Because data centers in cloud computing have begun to widely use virtualization technology, exploring a VM placement strategy for low energy consumption in a cloud data center provides a new research direction for improving energy efficiency in a data center. VM initial placement according to the optimal goal can be used to reduce energy consumption and reduce the number of active servers after the completion of cloud computing platform in a certain degree, and it has a positive effect on the operation and subsequent optimization for VM consolidation of the cloud computing platform.

In this paper, we use binary particle swarm optimization (BPSO) to implement VM initial placement with comprehensive consideration of CPU and disk, then we use model of energy efficiency fitness to adjust each particle's velocity and

displacement within the scope of certain iterations, update each particle's VM placement according to consideration of the globally optimal displacement and locally optimal displacement, and finally we can get a best VM placement which has the minimum energy consumption.

The rest of this paper is organized as follows: Section 2 discusses related work of VM placement. Section 3 discusses proposed model of energy efficiency fitness. Section 4 discusses VM placement strategy based on BPSO. Section 5 contains details of our algorithm. Section 6 is experiments. At last, Section 7 is the summary of this paper.

2. Related Work

With the rapid development of cloud computing, data centers also expand, as well as the energy consumption caused by data centers, which results in high operation costs. Therefore, reducing energy consumption of data centers is a problem that needs to be solved urgently. The energy consumption in a data center has a close relationship with utilization of resources; thus we should improve the utilization of resources to reduce energy consumption in order to reduce operation cost of data centers. At present, the most effective method to achieve the goal is to use virtualization technology, which can make better use of server resources.

Improving utilization of resources by using virtualization technology is mainly divided into two types: initial placement [2, 3] and dynamic VM consolidation [4, 5]. A lot of research on VM placement has been done by domestic and overseas scholars. Many studies have modeled VM placement as packing problems [6, 7]. The purpose of optimization of VM placement is to use the least amount of physical resources to meet demands of all virtual machines, which can improve efficiency and reduce energy consumption. Many works consider load condition or resource competition in terms of servers of CPU, memory, disk, network bandwidth, and so on. Because various factors should be considered to obtain the optimal solution of multiple dimensions, VM placement has been proved to be an NP-hard problem. The most basic method of VM placement is to use heuristic algorithms, such as best fit algorithm and first fit algorithm. Beloglazov and Buyya [8] have put forward MBFD (modified best fit decrease) algorithm which is used to consolidate virtual machines; the main idea of this literature is reallocating virtual machines which need to be migrated to servers to minimize the energy consumption. These simple methods fall into locally optimal solution because of less thought, resulting in extra waste of resources and increasing SLA violation, which has an effect on QoS as well. Therefore, we need to improve or replace heuristic algorithm.

Chen et al. [5] have described VM placement as a stochastic optimization problem. However, this approach is too simple, which only considers CPU and does not consider other resources. Li et al. [9] have proposed heuristic binary search algorithm, which places virtual machines in a few number of servers as far as possible to reduce energy consumption of the whole data center, and virtual machines of the same tenant are placed in the same server that can reduce the network consumption. The practicality of this algorithm is not good because only one resource type is considered.

Some scholars adopt intelligent algorithm to solve the problem of VM placement. Xu and Fortes [10] have used genetic algorithm to solve the problem of VM placement, but this method does not consider the overhead of VM migration, so it does not have good practicability. Li et al. [11] have proposed a strategy for VM placement that is based on multiobjective genetic algorithm; however the strategy cannot be applied to solve the problem of power consumption. In addition, it does not combine resource control and energy consumption. Farahnakian et al. [12] have used ant colony algorithm for VM consolidation, which can ensure the relatively better original performance and reduce energy consumption at the same time, but this algorithm is too complex, and the speed of its convergence is slow; therefore it is difficult to be implemented.

In recent years, particle swarm optimization (PSO) algorithm has drawn more and more attention from the researchers, because it is similar to genetic algorithm in terms of function, and it has many other advantages: quicker convergence, less parameters, easier operation. In addition, BPSO can be applied to solve the problem of VM placement. In this article, we present an energy-aware strategy for VM initial placement based on BPSO. We first design a multiresource model of energy efficiency fitness and then use BPSO to place virtual machines based on this model. After comparing BPSO with some heuristic algorithms and GA through experiments, we obtained the experimental results, which show that our proposed VM placement strategy not only effectively reduces energy consumption of servers, but also reduces the number of active physical machines as well as the proportion of SLA violation in a certain extent.

3. Energy Efficiency Fitness Model

3.1. Fitting Distance. Every cloud data center will hold the pool of servers, where there are a large number of physical machines. And each physical machine can load virtual machines of different numbers and different specifications; task requests from users and applications are sent by virtual machines. In a data center, utilization of resources such as CPU, disk, memory, bandwidth has an important implication to servers. Because user requests and the sizes of applications are different, their consumption of different types of resources varies, some requests need to consume more CPU resources, and some tasks need more disk utilization, while others consume more bandwidth. In this article, we consider two types of resources: CPU and disk.

Current CPU utilization in a server can be achieved by calculating the sum of CPU utilization in virtual machines running on the server. Similarly, utilization of disk in a server can be achieved by calculating the sum of used disk in virtual machines running on the server. The utilization of CPU and disk affects the energy consumption. Srikantaiah et al. [13] have studied the relationship between energy consumption and utilization of resources. In the literature, results of many

experiment shows that, when CPU utilization rate of a server reaches 70% and disk utilization rate reaches 50% at the same time, servers can not only get the minimum value of energy consumption, but also ensure a good performance. Therefore, in this literature, 70% and 50% can be seen as the optimal point for CPU utilization rate and disk utilization rate, respectively, in a server. Two values in the optimal point here are different from that considered separately, so we need to combine two parameters together when solving the problem of VM placement.

Next, we set up a fitness model. We put forward fitness distance Fit_k (k denotes k servers) to analyze the fitness of servers quantitatively; fitness distance can be gotten from Euclidean distance of resource utilization between current server and the optimal point. The function is shown as follows:

$$\mathrm{Fit}_k = \begin{cases} \sqrt{\sum_{n=1}^{N} \left(S_{kn} - Sbest_n\right)^2} & \text{others} \\ 0 & S_{kn} = 0,\ n = 1, 2, \end{cases} \tag{1}$$

where S_{kn} is nth resource utilization rate in server k, $Sbest_n$ represents optimal utilization of nth resource, and n is the number of resource types; since CPU and disk are considered here, n is 2. Fit_k is the fitness distance of server k, it is a quantitative representation of energy efficiency fitness. When a server is placed with a running VM, Fit_k is the Euclidean distance of resource utilization from the optimal point in server k. When a server is an idle server, which means no VM is deployed on this server and its fitness distance is 0, then energy efficiency fitness of the whole data center is shown as follows:

$$\mathrm{Fit} = \sum_{k} \mathrm{Fit}_k. \tag{2}$$

It represents the sum of Euclidean distance in each server. Because the model refers to optimal point [13] and this adaptive point has confirmed its high energy efficiency through many experiments, it is of great value to VM placement by using BPSO. And due to the availability of resource parameters, the model is of high practicability.

3.2. Problem Formulation. Since the value of energy efficiency fitness in a whole data center is closely related to VM deployment in servers, to obtain the lowest value of energy efficiency fitness, a strategy for VM placement is proposed to make CPU utilization rate and disk utilization rate of each server with one or more virtual machines run closer to optimal point to a great extent; then we can achieve best energy efficiency in the whole data center. The following is problem formulation.

Suppose that a data center has N virtual machines that need to be deployed on M servers ($k \in K$, $i \in I$, where K is the set of servers and I is the set of virtual machines), and virtual machines in each server need enough resources to keep a stable state. S_i^n denotes nth resource demand of VM i, and S_k^n denotes the capacity of resource n in server k. And resources of each server are limited; to guarantee resource utilization of each server in a proper range and ensure that the efficiency of each server is not too bad, we define a set of high and low threshold values of resource utilization: $U_n^{\mathrm{high}n}$ represents the maximum utilization of resource n in a server that hosted one or more virtual machines; on the contrary, U_n^{low} represents the minimum utilization. The two thresholds play a decisive role in VM placement. Once the value of resource utilization is no longer within these two thresholds, virtual machines need to be migrated. Then we define two binary variables: X_{ik} represents the existence of VM i in server k, and 1 denotes that VM i is placed on server k; Y_k decides server k is idle or not, and if the value is 0, that means server k is in the idle state and needs to be adjusted to obtain a state of low energy consumption. The goal of this article is to obtain the minimum energy efficiency fitness; the formula is as follows:

$$\begin{aligned} \text{Min} \quad \mathrm{Fit} &= \sum_{k=1}^{M} \mathrm{Fit}_k \\ &= \sum_{k=1}^{M} \sqrt{\sum_{n=1}^{2} \left(\frac{\sum_{i=1}^{N} X_{ik} S_i^n}{S_k^n} - Sbest_n \right)^2} \end{aligned} \tag{3}$$

and constraints are as follows:

$$\begin{aligned} &\sum_{i=1}^{N} X_{ik} = 1 \quad \forall i \in I \\ &U_n^{\mathrm{low}} \cdot S_k^n \le \sum_{i=1}^{N} X_{ik} \cdot S_i^n \le U_n^{\mathrm{high}} \cdot S_k^n \quad \forall k \in K \\ &X_{ik}, Y_k \in \{0, 1\} \quad \forall i \in I,\ \forall j \in J \\ &\qquad\qquad\qquad\qquad\qquad\qquad n = 1, 2, \end{aligned} \tag{4}$$

where $Sbest_n$ is the optimal point mentioned in the previous section.

4. Energy-Aware VM Initial Placement Strategy Based on BPSO

4.1. Introduction to BPSO. PSO is an intelligent optimization algorithm which was proposed originally by Eberhart and Kennedy according to the laws of some animals' food-searching [14, 15]. Subsequently, Zhang et al. improved this algorithm, added a dynamic weight to prevent too fast convergence which could lead to locally optimal solution, and then added the exploration of the PSO algorithm [16]. Compared to ant colony algorithm [12, 17, 18], genetic algorithm [7, 19, 20], and many other intelligent algorithm, PSO algorithm has fewer parameters, faster convergence, simpler code, easier operation, and so on, so it is of high practicability.

Standard PSO algorithm is only suitable for solving problems of continuous space; however, the problem of VM placement is a discrete optimization problem, so this algorithm cannot be directly applied to it. Improvement of PSO algorithm is necessary, and this improvement can be applied to solve optimization problems in the discrete space.

Then Kennedy and Eberhart designed corresponding binary version of PSO (BPSO) in 1997 [21], which is used to solve optimization problems in discrete space.

BPSO is similar to the original PSO; the difference is that its particles are composed of binary code. Each particle represents two variables: velocity and displacement. Each particle also stores the two values after each round of iteration: its own locally optimal location (L_i) and globally optimal location of the whole particle swarm (G). The two values are used to provide the basis of changes for the particle's velocity; the formula is as follows:

$$V_{id}(t+1) = \omega \cdot V_{id}(t) + c_1 \cdot r_d() \cdot \left(L_{id} - X_{id}(t)\right) + c_2 \cdot r_d() \cdot \left(G_d - X_{id}(t)\right). \quad (5)$$

ω is a dynamic weight, which is used to adjust the speed of convergence. And c_1 and c_2 represent learning factors, which can be called acceleration constants as well, both of which are usually set to 2; $r_d()$ is a random number, its value is in the range of $[0, 1]$. L_{id} and G_d are, respectively, locally optimal and globally optimal locations of server d in particle i. In order to guarantee the probability that the value of each bit in displacement is only 1 or 0, the sigmoid function is adopted:

$$S\left(V_{id}(t+1)\right) = \frac{1}{1 + \exp\left(-V_{id}(t+1)\right)}. \quad (6)$$

Particles change their value by

$$X_{id}(t+1) = \begin{cases} 1 & \text{if } r_d() \le S\left(V_{id}(t+1)\right) \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

4.2. VM Initial Placement Based on BPSO. Since BPSO is suitable to solve discrete optimization problems and has advantages over other algorithms, we use it to solve the problem of VM initial placement. In order to better use BPSO, we need to describe the structure of parameters and operation instructions, which can make the strategy of VM initial placement clearer. BPSO has not been used to solve the problem of the energy consumption in cloud data centers by now, so an improved BPSO algorithm is proposed in this article to deal with the problem of high energy consumption, which is a totally new idea.

4.2.1. Parameters Used by BPSO

① *Particle Displacement*. We define a particle swarm that has L particles; X_i^w is the displacement of particle w ($1 \le w \le L$), and i is the number of iterations. X_i^w is a N-bit vector:

$$X_i^w = \left(X_{i1}^w, X_{i2}^w, \ldots, X_{iL}^w\right). \quad (8)$$

Each bit corresponds to the status value of representative server; if one has a value of 1, it means corresponding server is in working status, and there is at least one VM in it; if the value is zero, the corresponding server does not have VMs and is in a state of low energy consumption.

② *Particle Velocity*. It is the same as particle displacement; V_i^w is the velocity of particle w. It is also a N-bit vector:

$$V_i^w = \left(V_{i1}^w, V_{i2}^w, \ldots, V_{iL}^w\right). \quad (9)$$

It decides whether the corresponding server of each bit needs to migrate VM or not. The initial state of each bit is 0.

③ *Locally Optimal Displacement and Globally Optimal Displacement*. We use Lb^w to define locally optimal displacement of particle w, which can get the best energy efficiency in particle w after iterations, and globally optimal displacement (Gb), which represents the best displacement of all particles that can lead to the best energy efficiency fitness so far. These two parameters play a critical role in changes of particle velocity; in other words, they determine the next step of VM deployment in particles through their locally and globally optimal displacement.

④ *Inertia Weight*. In particle swarm, if there is no inertia weight, the speed of convergence in BPSO will be greatly reduced, so inertia weight in BPSO plays a key role in searching results. However, too fast convergence will easily lead to locally optimal solution rather than globally optimal solution. And in different conditions, appropriate adjustment of inertia weight is extremely important. The start of BPSO carries out a global search ability, where a large inertia weight is needed to quickly find all solutions whose energy efficiency fitness is better than others. After the global search, BPSO should decrease the value of inertia weight and converts from global search to local search, which can get globally optimal solution in these good solutions, so dynamic inertia weight is necessary to improve the search ability of BPSO algorithm. The change rule of inertia weight is shown in

$$\omega = \omega_{\min} \cdot \exp\left[-\left(\frac{i}{\text{num}}\right)^p + 1\right]^r. \quad (10)$$

$\omega_{\min}$ is the lowest limit of inertia weight, i is the number of iterations, num is the total number of iterations, p is weight coefficient of changes, and r is the upper limit of weight adjustment. Overall, inertia weight gradually decreases by the increasing number of iterations, and, if needed, we can control the degree of urgency.

⑤ *Addition Operation and Subtraction Operation*. Since original formula of BPSO algorithm is a one-dimension formula, we need to define some mathematical operations. Addition operation is used to add those variables in the same dimension; similarly, subtraction operation is the subtraction of variables in corresponding dimension. For example, $(1, 0, 2, 1) + (2, 1, 0, 0) = (3, 1, 2, 1)$.

⑥ *Function Mapping*. Particle displacement needs a corresponding change according to particle velocity. Original BPSO is unlikely to converge to globally optimal particle, because if it converges to globally optimal particle, its velocity is zero, which will increase the possibility of changes in the corresponding bit; then the search will have more

randomness and lack of direction. And original mapping does not conform to the problem of VM placement, because if particle velocity can lead to VM migration, it does not necessarily result in changes of particle displacement. For example, server A hosts VM$\{1, 2, 3\}$, and server B hosts VM$\{4\}$; then the value of the status in server A and B is $(1, 1)$. If VM3 is migrated from server A to B, then server A hosts VM$\{1, 2\}$ when server B hosts VM$\{3, 4\}$; the status still keeps unchanged. We have to improve the sigmoid function.

The following are improvements:

$$V'_{id}(t+1) = \begin{cases} 1 - \dfrac{2}{1+\exp\left(-V_{id}(t+1)\right)} & V_{id}(t+1) \le 0 \\ \dfrac{2}{1+\exp\left(-V_{id}(t+1)\right)} - 1 & V_{id}(t+1) > 0. \end{cases} \quad (11)$$

$V'_{id}(t+1)$ is the probability that $X'_{id}(t+1)$ is 1, and then we use $X'_{id}(t+1)$ to change particle velocity. If $V'_{id}(t+1)$ is less than 0,

$$X'(t+1) = \begin{cases} 0 & r_d() \le V'_{id}(t+1) \\ X'_{id}(t) & \text{others.} \end{cases} \quad (12)$$

Otherwise,

$$X'_{id}(t+1) = \begin{cases} 1 & r_d() \le V'_{id}(t+1) \\ X'_{id}(t) & \text{others.} \end{cases} \quad (13)$$

$X'_{id}(t+1)$ is the transition of $X_{id}(t+1)$. Next is multiplication operation.

⑦ *Multiplication Operation*. Multiplication operation updates particle displacement and denotes multiplication operation, and the formula is

$$X(t+1) = X(t) \times X'(t+1). \quad (14)$$

Both $X'_{id}(t+1)$ and $X_{id}(t)$ can only be assigned two values: 0 and 1. When the value of $X'_{id}(t+1)$ is 1, we do not need to change the displacement of corresponding bit. On the contrary, if the value is 0, we need to observe the value of corresponding bit, namely, $X_{id}(t)$; if the value is 1, we need VM migrations; otherwise, the server can be used as a destination server, and virtual machines can be migrated to this server.

4.2.2. Updating VM Deployment. Based on the above, we can calculate particle velocity of each generation and determine the adjustment of particle displacement. When the transition of particle displacement takes 1, it means that there is no change in corresponding bit of particle displacement; on the contrary, if some bit of the transition is zero, corresponding bit of particle displacement needs to be adjusted, according to VM reallocation.

In this article, we use an updating strategy based on FFD (first fit decreasing), which can update VM displacement.

In order to get a better updating strategy, we need to adjust the server whose value of displacement is 0, and we get a new collection about the adjusted server, which has a two-dimension parameter.

We need to normalize these two parameters. We list all of virtual machines in $\overline{X}$ and form the collection $\mathrm{VM}_p\{\mathrm{VM}_{1p}, \ldots, \mathrm{VM}_{np}\}$. And each VM in VM_p has a normalized performance; its formula is

$$M_{ip} = \sqrt{\left(\frac{\mathrm{CPU}_{ip} - \mathrm{CPU}_p^{\min}}{\mathrm{CPU}_p^{\max} - \mathrm{CPU}_p^{\min}}\right)^2 + \left(\frac{\mathrm{DISK}_{ip} - \mathrm{DISK}_p^{\min}}{\mathrm{DISK}_p^{\max} - \mathrm{DISK}_p^{\min}}\right)^2}. \quad (15)$$

$\mathrm{CPU}_p^{\max}$ is the maximum CPU in VM_p, and $\mathrm{CPU}_p^{\min}$ is the minimum. Similarly, $\mathrm{DISK}_p^{\max}$ is the maximum disk of VMs in VM_p, and $\mathrm{DISK}_p^{\min}$ is the minimum.

According to this formula, we can receive a collection of normalized performance VM_p, then sort the element decreasingly, and, finally, leverage the normalized performance to reallocate VMs in VM_p based on BFD; then we will get new VM placement in VM_p. However, to prevent local optimum, we just use new VM placement in a certain probability.

We calculate two values of energy efficiency fitness according to formula (3), two values ($\mathrm{Fit}\,\overline{X}$ and $\mathrm{Fit}\,\overline{\overline{X}}$) denote the fitness of original VM placement and new VM placement in VM_p, respectively. The probability of using new VM placement in VM_p is as follows:

$$p = \frac{1/\mathrm{Fit}\,\overline{\overline{X}}}{\left(1/\mathrm{Fit}\,\overline{X}\right) + \left(1/\mathrm{Fit}\,\overline{\overline{X}}\right)} = \frac{\mathrm{Fit}\,\overline{X}}{\mathrm{Fit}\,\overline{X} + \mathrm{Fit}\,\overline{\overline{X}}}. \quad (16)$$

5. Algorithm

Data Structure

(1) *PS*[*pnumber*]: it represents the total number of the particle swarms.

(2) *Iter*: it represents the number of iterations in BPSO.

(3) *ServerList*: it is a list of servers and stores the information of each server, such as id, CPU and disk, the velocity and displacement, the transitions value, and the collection of virtual machines.

(4) *VMList*: it stores all information of virtual machines, such as the label and demands for CPU and disk.

(5) *Lbest*[*pnumber*]: it deposits optimal placement of each particle that obtains minimum energy efficiency fitness.

(6) *Lvalue*[*pnumber*]: it is the minimum energy efficiency fitness of current particle during the current iteration.

(7) *Gbest*: it is the optimal placement of the particle swarm that obtains minimum energy efficiency fitness.

```
Input: VMList, pnumber, Iter, ServerList
Output: Gbest
(1) Gvalue = ∞; Gbest = null;
(2) for i in pnumber
(3)    ServerList = RandomVMPlacement(VMList);
(4)    PS[i] = ServerList;
(5)    Lbest[i] = ServerList;
(6)    Lvalue[i] = Fitness(ServerList);
(7)    if (Gvalue > Lvalue[i]) then
(8)       Gbest = Lbest[i];
(9)       Gvalue = Lvalue[i];
(10)   end if
(11) end for
(12) while Iter > 0
(13)    for i in pnumber
(14)       Lbest[i] = BPSO(ServerList, Lbest[i], Gbest);
(15)       Lvalue[i] = Fitness(ServerList);
(16)       if (Gvalue > Lvalue[i])
(17)          Gbest = Lbest[i];
(18)          Gvalue = Lvalue[i];
(19)       end if
(20)    end for
(21)    Iter = Iter − 1;
(22) end while
(23) return Gbest;
```

Algorithm 1: VM initial placement based on BPSO.

(8) *Gvalue*: it is the minimum energy efficiency fitness of the particle swarm during the current iteration.

The pseudocode for our proposed strategy is presented in Algorithm 1. Algorithm 1 describes the overall process of initial VM placement. First is the generation of *pnumber* particles, which are *pnumber* server lists based on given size of the particle swarm. Each server list has the same servers, but their deployment of virtual machines is not identical because of randomness. Then according to *pnumber* kinds of VM placement, we determine the globally optimal placement and locally optimal placement which can get the least value of energy efficiency fitness. These complete the preparation for the implementation of BPSO (lines 3–11).

Next, in the case of finite iterations, we need to use BPSO algorithm for each particle, which can update its optimal VM placement and globally optimal placement of the whole particle swarm (lines 12–22). After the iterations, the globally optimal VM placement of the particle swarm is the VM placement we want to get (line 23).

Algorithm 2 is the algorithm of random VM placement. Each virtual machine randomly selects a server; if the server has sufficient resources to accommodate a virtual machine, then it will be placed in a server. When the server does not host other VMs before placing this virtual machine, then its displacement is set to 1 (the initial value is 0), to indicate that the server has one or more virtual machines. If current server does not have enough resources for this virtual machine, the algorithm will continue to look for the next server, until finding the right server.

```
Input: VMList Output: ServerList
(1) for vm in VMList
(2)    for server in ServerList
(3)       if (server has enough resources to hold vm) then
(4)          if (server.isEmpty()) then
(5)             server.setDisplacement(1);
(6)          end if
(7)          server.add(vm);
(8)          break;
(9)       end if
(10)   end for
(11) end for
(12) for server in ServerList
(13)    server.setVelocity(0);
(14) end for
(15) return ServerList;
```

Algorithm 2: Random VM placement.

Algorithm 3 is the operation of updating each particle during the iterations in BPSO. First get particle velocity and particle displacement for corresponding arithmetic operations. It updates the velocity and displacement of each server (each bit of particle velocity and particle displacement represents the velocity and particle of a server, respectively; initial value of each server's velocity is 0), and then it changes corresponding transition value according to updated velocity (lines 3–6). Transition value determines whether the deployment of current server needs to change or not (lines 7–15). All servers whose transition value is 0 join a new list, which need to redeploy the placement of virtual machines and calculate the energy efficiency fitness of servers in this list. Based on the normalized parameter, the algorithm uses BFD to redeploy VMs in this list and calculates its fitness. Comparing to the fitness before redeployment, if the value of this new fitness is smaller, then it updates *ServerList* in the probability gotten from formula (16).

6. Experiment Evaluation

6.1. Experimental Setup. In order to assess the advantages and disadvantages of proposed strategy, we carry out various simulated tests to test performance. The results of performance are compared with those of other heuristic algorithms, which can objectively show how good or bad our proposed strategy is based on BPSO. In addition to these heuristic algorithms, we use the idea of GA algorithm in articles [10, 11] and design the experiments to compare it with our proposed algorithm. In this paper, all experiments are conducted on the same computer, whose processor is AMD A6-3400-m APU with Radeon HD Graphics, memory is 6 G, and operating system is Windows 7 Professional SP1, and simulation experiments are conducted using the platform of eclipse, whose version is Mars.1 Release (4.5.1). In order to ensure the practicability of the algorithm, the configuration of servers in the experiment refers to the configuration of servers on the market.

Input: *ServerList*, *Lbest*[*i*], *Gbest*
Output: *Lbest*[*i*], *Gbest*
(1) **for** *server* **in** *ServerList*
(2) $X = server$.getDisplacement();
(3) $V = server$.getVelocity();
(4) Lx = Lbest[i].getServer(server.getId()).getDisplacement;
(5) Gx = Gbest.getServer(server.getId()).getDisplacement;
(6) $V = \omega \cdot V + c_1 \cdot rd(Lx - X) + c_2 \cdot rd() \cdot (Gx - X)$;
(7) **if** $(V \le 0)$ **then**
(8) $V' = 1 - \dfrac{2}{1 + \exp(-V)}$
(9) **else**
(10) $V' = \dfrac{2}{1 + \exp(-V)} - 1$
(11) **end if**
(12) X' =*server*.getTransition();
(13) **if** $(V' < 0$ and $r_d() \le V')$ **then** $X' = 0$
(14) **else if** $(V' > 0$ and $r_d() \le V')$ **then** $X' = 1$
(15) **end if**
(16) **if** $(X' = 0)$ **then**
(17) *RedeployServerList*.add*(server)*;
(18) **end if**
(19) **end for**
(20) OrginalMigrationServerList = RedeployServerList;
(21) *RedeployServerList*.sortByBFD();
(22) **if** (Fitness(*OrginalMigrationServerList*)>Fitness(*RedeployServerList*)) **then**
(23) update the information of servers in *RedeployServerList*
(24) update the *VMList* in ServerList according to *RedeployServerList* in the probability gotten from formula (16);
(25) update *Lbest*[*i*], *Gbest* according to changed *ServerList*;
(26) **end if**
(27) **return** *Lbest*[*i*], *Gbest*;

Algorithm 3: BPSO.

At the start of experiments, VMs are assigned resources of different orders of magnitude according to the demands. In order to ensure that servers have enough resources to hold virtual machines, we decide that the numbers of servers and VMs are the same in each experiment.

The following is the analysis of experimental results.

① *Energy Efficiency Fitness*. First, we compare our proposed strategy to other four algorithms based on energy efficiency fitness. Figure 1 shows the energy efficiency fitness of five methods under the conditions that different numbers of servers and virtual machines are deployed. In this experiment, the population of BPSO is 50, and the number of iterations is 100. In order to eliminate the interference of other factors in the experiment, upper and lower thresholds are consistent.

The result from Figure 1 concludes that the proposed strategy based on BPSO can get the smallest energy efficiency fitness in all scales of servers and VMs, and compared to the other four algorithms, its control of energy efficiency fitness is very obvious, mainly because BPSO has an adjustment of globally optimal solution and locally optimal solution in each iteration, which can ensure less energy consumption. While three heuristic algorithms are simple greedy algorithms, which lack such adjustment, they can easily lead to larger energy efficiency fitness and more energy consumption; GA can get lower values as well than the other three heuristic algorithms; however, there is a high probability that the results it get may be a locally optimal solution, so its values of best fitness are higher than that gotten by our proposed algorithm.

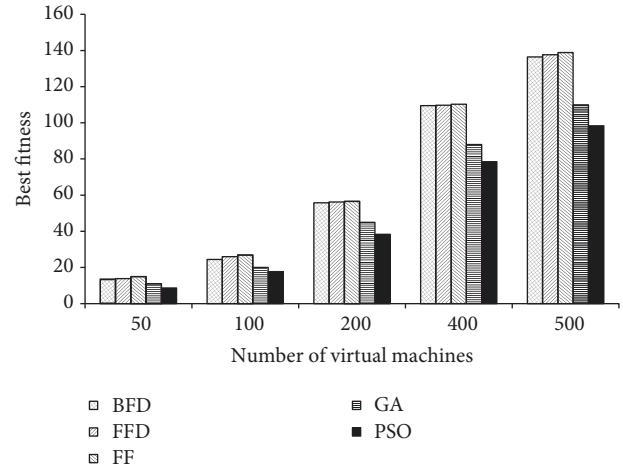

Figure 1: Energy efficiency fitness.

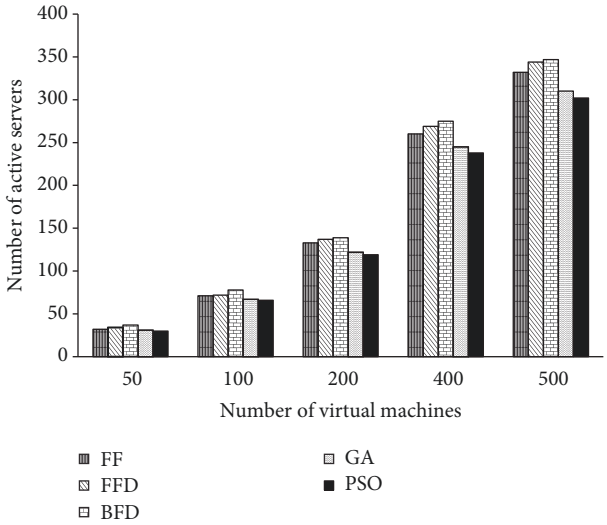

Figure 2: Number of active servers.

② *Number of Active Servers*. In addition to energy consumption, the number of active servers is also an important performance metric. We keep the parameters the same as what was set in the previous experiments and then compare numbers of active servers after initial VM placement based on five algorithms, respectively. Figure 2 is the result of experiments.

BPSO used in experiments has 100 populations and 100 iterations. And other parameters are consistent. Evidently compared to the other four algorithms, our proposed strategy based on BPSO can produce relatively fewer active servers, because our proposed strategy reallocates VM placement of abnormal servers by using BFD during iterations, which also ensures that the number of active servers cannot be too large; this also shows that proposed strategy can take advantages of other algorithms to improve superiority by itself; this also shows that proposed strategy outperforms the other algorithms.

③ *Parameter Setting*. According to BPSO algorithm introduced in this paper, some parameters need to be assigned suitable values. Next, we test the influence of BPSO caused by these parameters.

(A) High and Low Threshold of CPU and Disk. High and low threshold of CPU and disk may affect the status of VM placement; then we change them to observe changes of servers' energy efficiency fitness in this experiment, where the iterations are 200 and the number of VMs is 200. Figure 3 is the result of changes. We conclude from the figure that changes of energy efficiency fitness caused by two thresholds are not very evident; however there is a trend that when the high threshold is 0.85 and the low threshold is 0.1, we get the smallest value. And when the low threshold of CPU and disk is 0.1, values of energy efficiency fitness are smaller than those in other conditions.

(B) Iterations. Next we do a research on energy consumption influenced by iterations. According to the previous

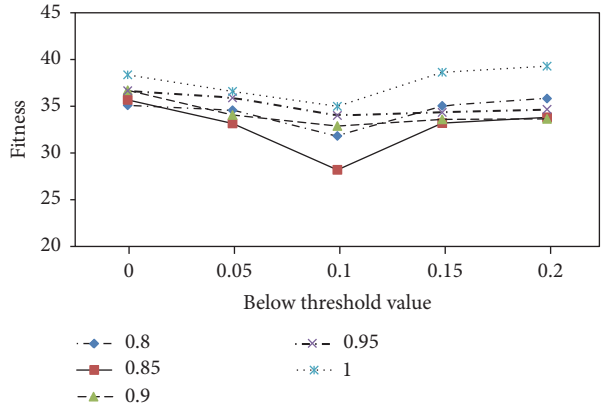

Figure 3: High and low threshold of CPU and disk.

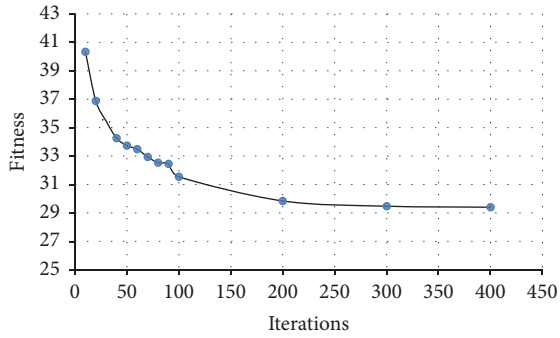

Figure 4: Iterations.

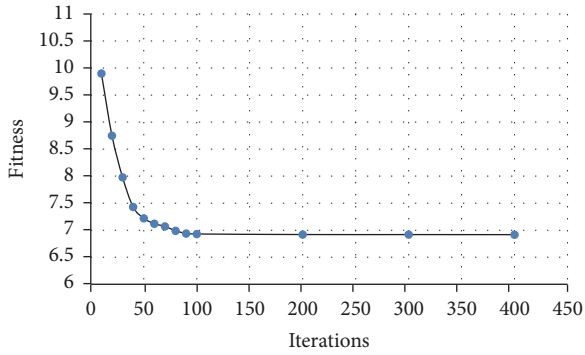

Figure 5: Iterations (50 virtual machines).

experimental results, 0.85 and 0.1 are set as the high and low thresholds of CPU and disk, respectively, and the number of generated virtual machines is 200, and the number of population is 50. Figure 4 shows that with the increase of iterations, the value of energy efficiency fitness is becoming more stable, and when the number of iterations is more than 200, the energy efficiency fitness is almost consistent.

The trend of changes of the fitness will not be affected when the number of VMs is changed. Figures 5 and 6 are the trends of energy efficiency fitness when a data center has 50 and 100 VMs, respectively.

Of course, changing the size of populations will not affect the trend of changes of fitness either. Figures 7 and 8

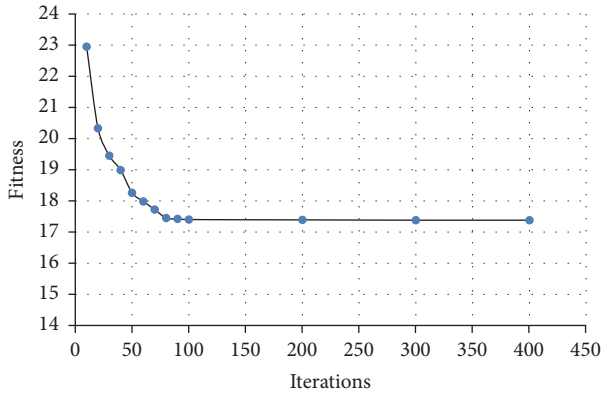

Figure 6: Iterations (100 virtual machines).

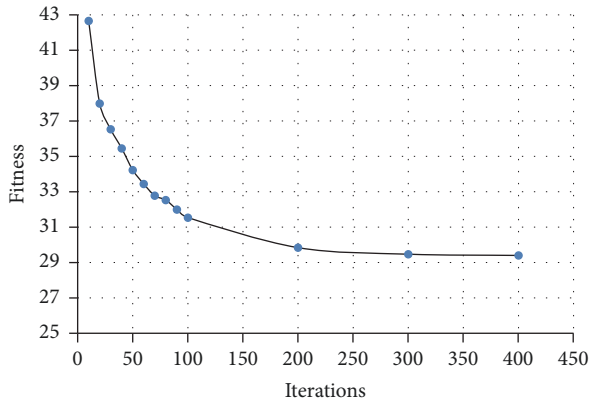

Figure 7: Iterations (30 populations).

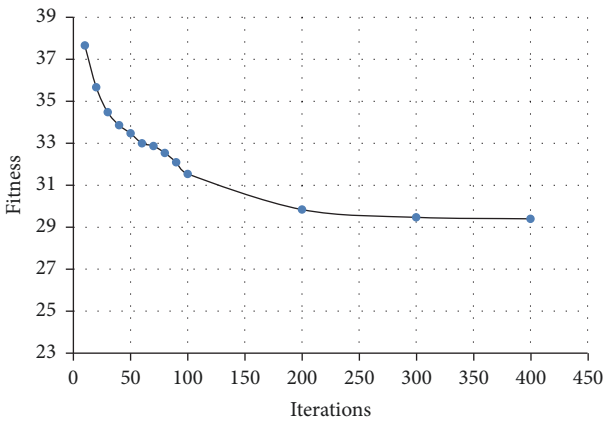

Figure 8: Iterations (100 populations).

are the trends of energy efficiency fitness when the size of populations is 30 and 100, respectively. With the increase of populations, the range of energy efficiency fitness is getting smaller.

(C) The Number of Populations. The number of populations is also an important parameter that influences energy consumption. If the size of populations is too small, it will affect the performance of BPSO, the convergence, and the value of energy efficiency fitness. Figure 9 is the result of

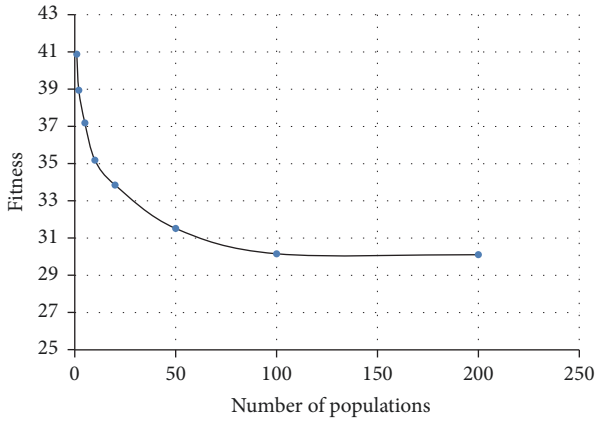

Figure 9: Number of populations.

energy efficiency fitness brought by different numbers of populations. The figure shows that energy efficiency fitness is in balance when the number of populations is more than 100.

To sum up, our proposed strategy based on BPSO is not only better than the other three heuristic algorithms (FF, FFD, BFD) and GA in terms of energy consumption gotten through value of energy efficiency fitness, but also better in other performances such as the number of active servers. In addition, this article finds through experiments that parameters in BPSO can affect the size of energy consumption, so we can adjust the various parameters to achieve optimal status of our proposed strategy.

7. Conclusion

With the rapid development of cloud computing, the demands of data centers are increasing. Larger scale of the cloud data center brings about enormous energy consumption, which cannot be ignored. Therefore, solving or alleviating energy problem becomes the urgency. VM initial placement is an important part of solution to reduce energy consumption. In this paper, we design energy-aware VM initial placement based on BPSO, which can obtain optimal VM placement that has a minimum energy consumption according to the adjustment of locally optimal placement and globally optimal placement. Through simulations and results, we can prove good performance of our proposed strategy based on BPSO. In addition, the influences of the performance caused by relevant parameters in BPSO are also discussed. In future work, we need to consider the best collocation of different parameters in detail and do further research about details of our proposed strategy based on BPSO.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is sponsored by Primary Research & Development Plan (Social Development) of Jiangsu Province (BE2017743) and National Science Foundation of China (no. 61602264).

References

[1] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and Distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.

[2] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting spatio-temporal tradeoffs for energy-aware MapReduce in the cloud," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1737–1751, 2012.

[3] C. C. T. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," in *Proceedings of the 25th IEEE International Conference on Advanced Information Networking and Applications, AINA 2011*, pp. 348–355, Singapore, March 2011.

[4] T. Cerling, J. Buller, C. Enstall, and R. Ruiz, *Mastering Microsoft® Virtualization*, Wiley Publishing, Inc., Indianapolis, IN, USA, 2009.

[5] M. Chen, H. Zhang, Y. Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 594–601, Dublin, Ireland, May 2011.

[6] B. Moores, "Autonomic virtual machine placement in the data center[C]//," in *Proceedings of the IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pp. 220–225, 2013.

[7] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh, "Toward virtual machine packing optimization based on genetic algorithm," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5518, no. 2, pp. 651–654, 2009.

[8] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the ACM 8th International Workshop on Middleware for Grids, Clouds and e-Science (MGC '10)*, pp. 1–6, Bangalore, India, December 2010.

[9] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1842–1850, can, May 2014.

[10] J. Xu and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," in *Ieee/acm International Conference on Green Computing and Communications & 2010 Ieee/acm International Conference on Cyber, Physical and Social Computing*, pp. 179–188, IEEE, Hangzhou, China, December 2010.

[11] Q. Li, Q. Hao, L. Xiao, and Z. Li, "Adaptive management and multi-objective optimization for virtual machine placement in cloud computing," *Chinese Journal of Computers*, vol. 34, no. 12, pp. 2253–2264, 2011.

[12] F. Farahnakian, A. Ashraf, T. Pahikkala et al., "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187–198, 2015.

[13] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing[C]//," in *Proceedings of the Conference on Power Aware Computing and Systems*, pp. 10-10, 2008.

[14] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micromachine and Human Science*, pp. 39–43, IEEE, Nagoya, Japan, October 1995.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimizatio," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, IEEE, Perth, Australia, December 1995.

[16] X. Zhang, Y. Du, Z. Qin, G. Qin, and J. Lu, "A Modified Particle Swarm Optimizer," in *Advances in Natural Computation*, vol. 3612 of *Lecture Notes in Computer Science*, pp. 592–601, Springer, Berlin, Germany, 2005.

[17] W. Qinghong U, J. Zhang H, and U. Xin He X, *AN ANT COLONY ALGORITHM WITH MUTATION FEATURES[J]. Journal of Computer Research*, 1999.

[18] P.-Y. Yin and J.-Y. Wang, "Ant colony optimization for the nonlinear resource allocation proble," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1438–1453, 2006.

[19] H. Wang, D. Lin, and M.-Q. Li, "A competitive genetic algorithm for resource-constrained project scheduling problem," in *Proceedings of the International Conference on Machine Learning and Cybernetics, ICMLC 2005*, vol. 5, pp. 2945–2949, August 2005.

[20] S. Jang H, T. Kim Y, and J. Kim K, "The Study of Genetic Algorithm-based Task Scheduling for Cloud Computing[J]," in *Proceedings of the International Journal of Control Automation*, pp. 157–162, 2012.

[21] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," vol. 5, pp. 4104–4108, 1997.