Soft Comput (2015) 19:2265–2273
DOI 10.1007/s00500-014-1406-6

METHODOLOGIES AND APPLICATION

Dynamic deployment of virtual machines in cloud computing using multi-objective optimization

Bo Xu · Zhiping Peng · Fangxiong Xiao · Antonio Marcel Gates · Jian-Ping Yu

Published online: 13 August 2014

Abstract Cloud computing is regarded as the fifth utility service and is the next generation of computation. The computing resources can be dynamically allocated according to consumer requirements and preferences Virtual machine deployment has an important role in cloud computing, and aims to reduce turnaround times and improve resource use. In essence, the deployment of virtual machines is a multi-objective decision problem that must consider key factors. That is, we need to optimize the resource use and migration times. In this paper, we propose the multi-objective comprehensive evaluation model for the dynamic deployment of virtual machines. We then use an improved multi-objective particle swarm optimization (IMOPSO) to solve the problem. We have designed two simulation experiments using the CloudSim toolkit: the first experimental results show that on comparison of our improved algorithm with the traditional single-objective algorithms PSO and QPSO, our method is feasible and efficient; the second experimental results show that IMOPSO can search effectively, maintain population diversity, and quickly converge to the Pareto optimal solution without losing stability. The obtained Pareto optimal solution set has a better convergence and distribution than a comparative method.

Keywords Virtual machine deployment · Particle swarm optimization · Multi-objective optimization · Cloud computing

1 Introduction

Cloud computing is regarded as the fifth utility service after water, gas, electricity, and telecommunication services, and is the next generation in computation (Buyya et al. 2009). One of the key aspects that makes cloud computing different from grid computing is that the computing resources (either hardware or software) are virtualized and allocated as services (Foster et al. 2008). The computing resources can be dynamically allocated according to consumer requirements and preferences (Armbrust et al. 2010).

Communicated by V. Loia.

B. Xu · Z. Peng (✉)
Guangdong Provincial Key Lab of Petrochemical Equipment Fault Diagnosis, Department of Computer Science and Technology, Guangdong University of Petrochemical Technology, Guangdong 525000, China
e-mail: pengzp@foxmail.com

B. Xu
e-mail: xubo807127940@163.com

B. Xu · F. Xiao
School of Software Engineering, South China University of Technology, Guangdong 510006, China

F. Xiao
School of Information and Statistics, Guangxi University of Finance and Economics, Guangxi 530003, China

A. M. Gates
Hawaii Pacific University, Honolulu, HI 96813, USA

J.-P. Yu
Key Laboratory of High Performance Computing and Stochastic Information Processing (Ministry of Education of China), College of Mathematics and Computer Science, Hunan Normal University, Hunan 410081, China

J.-P. Yu
High Technology Research Key Laboratory of Wireless Sensor Networks of Jiangsu Province, Nanjing University of Posts and Telecommunications, Jangshu 210003, China

A core issue of cloud computing is the deployment of virtual machine (VM) (Stillwell et al. 2010). The service providers simultaneously receive large numbers of computing requests, with different requirements and preferences from different users. Some users require less computing resources and lower cost, and others have higher requirements and take more computing resources (Kong et al. 2011). It is important to research techniques to deploy VM that meets the users' requirements while maximizing efficiency (Sindhu and Mukherjee 2011). The VM deployment problem is dynamic because users have different requirements that can change over time (Warneke and Kao 2011). Traditional algorithms make underlying assumptions that ignore important issues, and cannot meet the requirements of this dynamic environment. In such cases, the optimization algorithm must track a moving optimum as closely as possible, rather than simply finding a single good solution (Blackwell et al. 2006). So, the VM deployment problem can be classified as a multi-objective optimization problem. It is extremely challenging to quantify the performance of a VM deployment policy for different applications and service models under varying service level agreements, resource uses, and power consumptions (Chaisiri et al. 2012).

It has been argued that multi-objective evolutionary algorithms (MOEA) may be particularly suitable for this type of problem, and a large number of MOEA variants for dynamic optimization problems have been proposed over the past decade (Cruz et al. 2011). However, many of these approaches either cannot scale to the required size or are too fragile for a dynamic environment (Cruz et al. 2011). The development of new algorithms is challenging. The multi-objective particle swarm optimization (MOPSO) algorithm is a new algorithm that combines the principles of particle swarm optimization (PSO) and multi-objective optimization. It has been shown to perform better than conventional multi-objective algorithms for multi-objective optimization problems (Coello and Lechuga 2002).

The remaining parts of this paper are organized as follows. In Sect. 2, we discuss recent related work in the VM deployment problem. In Sect. 3, we describe the background of the VM deployment problem and MOPSO. In Sect. 3.2, we propose the multi-objective comprehensive evaluation model for VM dynamic deployment, which we then solve using improved multi-objective particle swarm optimization (IMOPSO). We present the experimental results and analysis in Sect. 4, and our conclusions in Sect. 5.

2 Related work

When considering future and recent conferences, journal special issues, and research reports, it becomes evident that there is growing interest in the VM deployment problem. We have identified some of the limitations reported in literature. Mahdavi et al. (2011) broke the VM deployment problem down into the packing and multi-objective optimization problems. First, they used a genetic algorithm to deal with the combinatorial optimization problem, and then they combined the technique with fuzzy logic to optimize the overall resource consumption, energy consumption, and overhead of heat dissipation. However, this approach does not consider the cost of VM migration. It considers static VM placement, but not dynamic deployment based on the migration of VM. Kourai focused on issues related to the shutdown of physical servers within the data center and the recovery of virtual machines and applications, using hierarchical models to optimize VM and application deployment (Kourai et al. 2011). Wilcox described VM deployment as a multi-objective optimization problem in cloud computing, mainly taking into account two goals: the number of physical servers used in the deployment of a VM, and VM migration times. They proposed a multi-objective genetic algorithm for VM deployment based on the classic NSGA-II algorithm (Wilcox et al. 2011). Aksaç created a two grade control management system to map a VM to a physical machine, using multi-objective optimization methods to solve potential conflicts in VM deployment (Aksaç et al. 2011). Sindelar formalized the problem as a multi-dimensional knapsack problem, handling deployment constraints as a separate one-dimensional problem (Sindelar et al. 2011). Maguluri considered the needs of users to be a random process and proposed a non-pre-emptive VM configuration policy for long framework times (Maguluri et al. 2012). This strategy can achieve near optimal system throughput. Maurer et al. (2012) proposed a VM deployment algorithm that used forecasting techniques and heuristic packing algorithms. They achieved the minimum number of physical machines and ensured a certain amount of service-level agreement. Takahashi et al. (2012) formalized the problem as a combination of multi-unit auction models. Nguyen et al. (2013) proposed a VM scheduling policy based on a genetic algorithm, which used historical data from the cloud computing system and the system's current state to achieve the load balancing and minimal overhead of VM migration. However, this strategy ignores the data center concerns of resource use and the related energy consumption. Sato et al. (2013) proposed a dynamic optimization of VM placement by predicting the resource usage. The proposed method predicts the future resource usage using an autoregressive model, but it must be evaluated on the servers that the VMs are deployed on.

The general issues related to VM deployment can be described as packing problems. The boxes of the packing problem are physical servers in the data center, the capacities of the boxes are server-related properties (such as CPU, memory, storage), and the items in the box cannot exceed a threshold according to the server properties (Hyser et al. 2007). The boxed items are VMs. The size of a VM is deter-

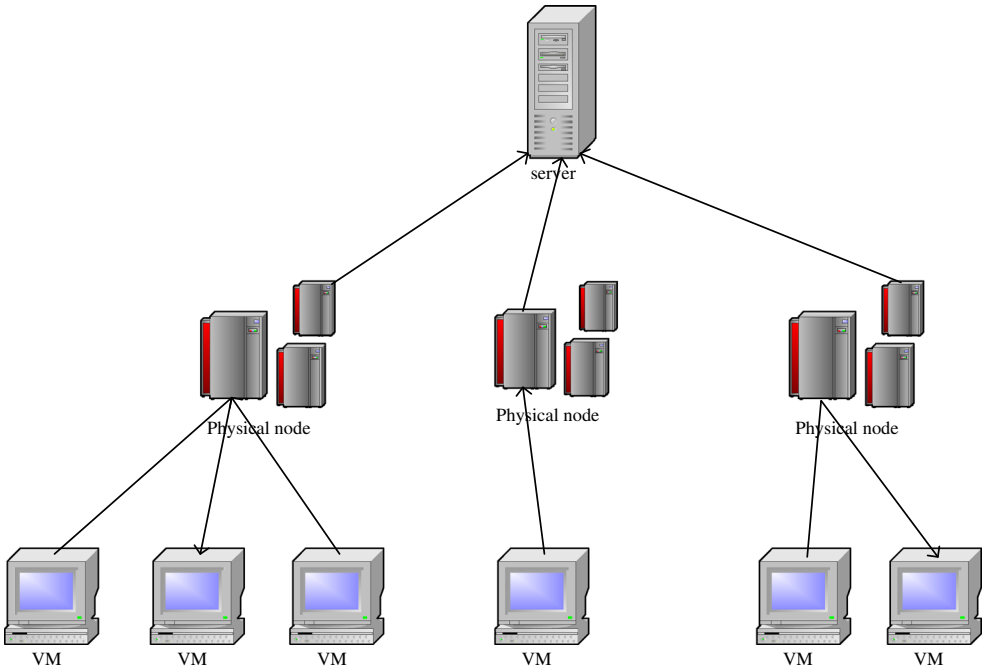

Fig. 1 VM deployment model

mined by the user's request because the data center offers a variety of VMs. VMs also have some properties similar to physical servers; using these properties we can determine whether the VM is suitable for deployment on a physical machine. However, there are some differences between the packing problem and VM deployment. For example, the VM requests arrive at different times. The developments of heat transfer methods suggest that VM deployment can be more flexible (Mohammadi et al. 2011). The typical VM deployment model is shown in Fig. 1:

Suppose there are M virtual machines that need to be deployed on N physical servers. There are N^M possible deployments, so this is an NP intricate packing problem. We need to find a set of approximate optimal solutions and apply a heuristic algorithm that has been used to find the approximate solution of optimization problems for a wide range of applications.

In this paper, we propose an improved multi-objective particle swarm optimization (IMOPSO) to solve the VM dynamic deployment problem in a cloud computing environment. The aim of the optimization method is to converge to the Pareto optimal solution as fast as possible, without losing stability.

3 IMOPSO for virtual machine deployment

3.1 Multi-objective particle swarm optimization (MOPSO)

PSO was derived from complex adaptive systems (CAS) (Eberhart and Kennedy 1995). CAS theory was proposed in 1994. A CAS member is called a body. For example, in bird research systems, each bird is called the body. A body has adaptability; it can communicate with the environment and other bodies, change its structure and behavior in accordance with the process of "learning", or "accumulate experience". PSO was first formally proposed in 1995 by Eberhart and Kennedy, and was originally derived from studies on the foraging behavior of birds (Eberhart and Kennedy 1995). The PSO algorithm aims to find the global optimum value. PSO converges fast, uses simple operations, and requires few parameters. The basic PSO algorithm speed formula is as follows (Eberhart and Kennedy 1995):

$$V_{id}(t+1) = V_{id}(t) + r_1 * c_1 * (P_{id}(t) - X_{id}(t)) + r_2 * c_2 * (P_{gd}(t) - X_{id}(t)) \quad (1)$$

$$X_{id}(t+1) = V_{id}(t) + X_{id}(t) \quad (2)$$

where $V_{id}(t+1)$ denotes the speed value of i particle d dimension in the $t+1$ generation r_1 r_2 is random number in [0, 1], c_1, c_2 is the velocity coefficient and is a constant, P_{id} is the individual current best location, P_{gd} is the current global best position, and $X_{id}(t)$ is the position of i particle d dimension in t generation. PSO is a natural candidate for multi-objective optimization because it is relatively simple and is a population-based technique (Reyes-Sierra and Coello Coello 2006). Since Parsopoulos and Vrahatis (2002) presented the first study of the performance of PSO in multi-objective optimization problems, many different MOPSOs have been reported, for example (Heo et al. 2006; Janson et al. 2008).

3.2 Multi-objective comprehensive evaluation model

VM deployment is in essence a multi-objective decision problem. The decision must be based on some key factors. That is, we need to optimize the resource use and migration times.

3.2.1 Resource use

In cloud computing, resources mainly refer to the physical server resources. The aim of resource use is to waste the least amount of resources. There are different ways to deploy VMs, which can result in very different resource uses. When user requests for VMs are predictable, data centers can reserve appropriate resources for the next stage. However, if the predictions deviate from the truth, there is a tremendous waste of resources. We do not consider predictable resource requests in this article and instead consider that user requests and the system's current state control the deployment of the VMs.

3.2.2 Migration times

Too much migration results in a large overhead, which wastes a lot of the available resources. In addition, the generated loss caused by migration times may result in a violation. Each data center has a column of physical servers, and each physical server can host multiple VMs. We have assumed that the physical attributes of the physical servers are fixed and that a VM can be migrated between any two physical machines. Under normal circumstances, the VM dynamic deployment framework is shown in Fig. 2.

Figure 2 shows that the VM's dynamic deployment is combined with both the current user's requests and the deployment status of VMs that already exist in the data center. The algorithm is used to determine a target physical machine, decide if a VM should be migrated, and deploy new VMs.

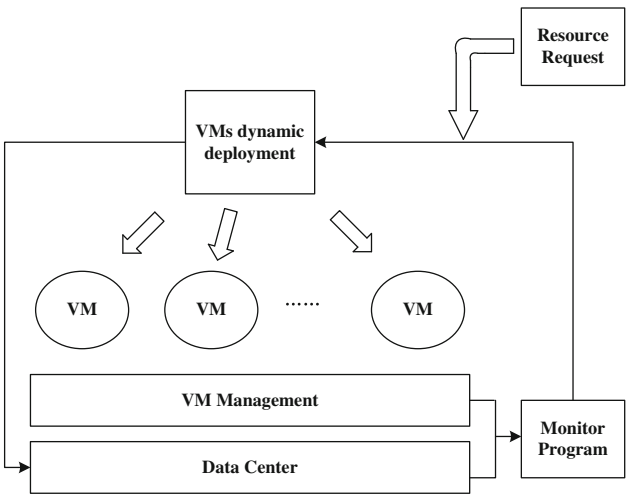

Fig. 2 Deployment framework for VMs

In the given framework, this paper uses MOPSO for the VM deployment. First, we define the relevant parameters, assuming that there are N physical servers that can provide virtualization services. The total number of VMs is $m = m_r + m_s$, where m_r represents the newly requested quantity of VMs, and m_s is the number of VMs already in the data center. In this article, each physical server's resources are represented by the triplet $R_i = \{R_i^C, R_i^M, R_i^B\}$, $i \in \{0, 1, ...N\}$, where R_i^C represents the number of processing units of CPU, R_i^M represents the size of memory, and R_i^B represents the size of bandwidth. Similarly, each VM can be expressed using $V_j = \{V_j^C, V_j^M, V_j^B\}$, $j \in \{0, 1, ...m\}$. We use η_{ij} to represent whether a VM is deployed to the server. If $\eta_{ij} = 1$, representing the i-th VM is deployed to the j-th server; if $\eta_{ij} = 0$, it is not deployed there. Thus, a parameter matrix of deployment for all VMs is

$$\eta = \begin{bmatrix} \eta_{11} & \eta_{12} & \cdots & \eta_{1N} \\ \eta_{12} & \ddots & \cdots & \eta_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{m1} & \eta_{m2} & \cdots & \eta_{mN} \end{bmatrix}. \tag{3}$$

The objective and constraint functions for the optimization are

$$\max \varphi_1(R_i, V_j) = \frac{\frac{\sum_{j=1}^{m} V_j^C}{\sum_{i=1}^{N} R_i^C} + \frac{\sum_{j=1}^{m} V_j^M}{\sum_{i=1}^{N} R_i^M} + \frac{\sum_{j=1}^{m} V_j^B}{\sum_{i=1}^{N} R_i^B}}{3} \tag{4}$$

and

$$\min \varphi_2(m_i) = \sum_{j=1}^{m} m_j, \tag{5}$$

such that

$$\begin{aligned} &\sum_{i=1}^{N}\sum_{j=1}^{m} \eta_{ij} = 1 \\ &\sum_{i=1}^{N}\sum_{j=1}^{m} \eta_{ij} V_j^C < R_j^C \\ &\sum_{i=1}^{N}\sum_{j=1}^{m} \eta_{ij} V_j^M < R_j^M \\ &\sum_{i=1}^{N}\sum_{j=1}^{m} \eta_{ij} V_j^B < R_j^B \end{aligned} \tag{6}$$

Equation (4) represents the maximum resource use, and Eq. (5) represents the minimum number of VM's migrations. The constraints in Eq. (6) show that a VM can only be deployed on one server, and that when multiple VMs

are deployed the VM's resources cannot exceed the resource capacity of the physical server.

3.3 Improved algorithms

PSO is used to simulate the behavior of flying foraging birds, through collective cooperation among birds so that the whole flock of birds is optimal. The basic idea of particle swarm optimization is to abstract individual birds into a particle without volume and mass, where each particle represents a potential solution; through the particle collaboration and information sharing to find the optimal solution, each particle can via certain rules adapt to the estimated value of its own position, each particle can remember the best location found currently known as locally optimal "l-Opt", and also remember the best location of all particles in the population found, called global optimum "g-Opt".

We increased the inertia weight w, using the velocity and position formulas. The value of w monotonically decreases; when it is large it broadens the scope of the search and when it is small it encourages convergence. We calculated the velocity and position based on the improved formulas for particle velocity and position and then calculated the fitness of the particle. However, it is not enough to increase w, because although this approach converges faster, it can easily fall into a local optimum. When the historical individual optimal value is equal to the current optimum, or it is equal to the current best, we consider that this particle may be trapped in a local optimum and randomly select the position of the particle. This can effectively reduce premature convergence and greatly increase the probability of obtaining the global optimum. As the PSO algorithm avoids getting trapped in a local optimum, we have modified the speed formula to

$$v_{id}(t+1) = w_{t+1} * V_{id}(t) + r_1 * c_1 * (P_{id}(t) - X_{id}(t)) + r_2 * c_2 * (P_{gd}(t) - X_{id}(t)), \quad (7)$$

$$X_{id}(t+1) = V_{id}(t) + X_{id}(t), \quad (8)$$

$$w_{t+1} = w_t - w_t * \left(\frac{t+1}{T_{\max}}\right), \quad (9)$$

where w_{t+1} is the inertia weight at $t+1$. Equation (9) guarantees that the inertia weight w monotonically decreases. In the formula, $i = 1, 2, \ldots, m$, $n = 1, 2, \ldots, N$, k is the iteration number, and r_1 and r_2 are the random numbers in [0,1]. Both the random numbers can maintain the diversity.

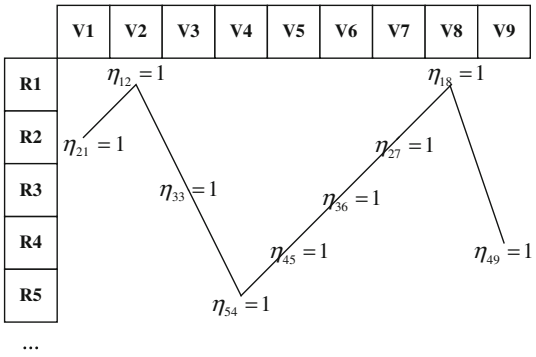

Fig. 3 Deployment scenarios for a VM resource

c_1, c_2 is the learning factor. In an N-dimensional search space, $X_i = (X_{i1}, X_{i2}, \ldots X_{id}...)$ represents the i-th particle position vector of N-dimensional $V_i = (V_{i1}, V_{i2}, \ldots V_{id}...)$ represents the i-th particle speed, $P_i = (P_{i1}, P_{i2}, \ldots P_{id}...)$ represents a local optimum position searched by the i-th particle, and $P_g = (P_{g1}, P_{g2}, \ldots P_{gd}...)$ represents the global optimal position searched by the i-th particles.

3.4 The IMOPSO for VM deployment

The VM deployment problem is a multi-objective optimization problem. Combinatorial optimization problems can be solved by simple enumeration, but for huge scales this is unrealistic, because when the problem size is exponentially large, the accuracy of the global search method for the optimal solution is almost impossible. Therefore, it cannot be solved in polynomial time. Heuristic algorithms can effectively deal with multi-objective combinatorial optimization problems.

PSO is appropriate given the theoretical basis of the problem, but general PSO optimizes a single objective. In this paper, we have adapted PSO for multi-objective optimization. A PSO algorithm for deploying VM resources is presented below.

Figure 3 shows a VM deployed on a server. In this example, the deployment parameter matrix in Eq. (3) has the entries $\eta_{21} = \eta_{12} = \eta_{33} = \eta_{54} = \eta_{45} = \eta_{36} = \eta_{27} = \eta_{18} = \eta_{49} = 1$. This node of the cloud computing resource path has the order R_2, R_1, R_3, R_5, R_4, R_3, R_2, R_1, R_4 .

We propose the following algorithm for achieving the best resource allocation of VMs in the cloud.

Algorithm1. The IMOPSO Algorithm

Given an initial size of population X, randomly generate position $x_i = (x_{i1}, x_{i2}, \cdots x_{iN})$ and speed $v_i = (v_{i1}, v_{i2}, \cdots v_{iN})$. The judging threshold and the maximum number of iterations $t_{\max}$ are also given;

For each iteration t in [1, $t_{\max}$] **do**

Use Equation (4) (5) to calculate the objective function $\varphi_1(R_i, V_j)$ and $\varphi_2(m_i)$;

From the resulting local optimal values, select two global optimum g-Opt [1] and g-Opt [2];

Calculate g (the average of two global optimal values), and the Euler distance $dg = \sqrt{\sum_{i=1}^{2}(g\_Opt[i] - g)^2}$ of the global optimum;

Calculate the Euler distance $dl[i]$ of the local optimum of each particle;

If $dl[i] < dg$, select a local optimum value from l-Opt [1, i] and l-Opt [2, i]; if $dl[i] > dg$, set the average of l-Opt [1, i] and l-Opt [2, i] as the new local optima;

Use the obtained global and local optimal values to update each particles velocity and position;

end for

If the stopping criterion is satisfied, output the results.

4 Experimental results and analysis

The CloudSim toolkit can model cloud computing systems and the behaviors of components such as VMs, data centers, and resource provisioning policies (CloudSim 2009; Rodrigo et al. 2011). We have conducted experiments to verify the effectiveness of the proposed IMOPSO for VM deployment. The system used Windows XP SP3, MyEclipse version 8.5 and the JDK version jdk1.6.0. We used CloudSim version CloudSim-3.0 and the Ant build tool Ant 1.8.1. We implemented the IMOPSO algorithm using an inherited class in the basic CloudSim class.

We have designed two simulation experiments; the first experiment is to compare our improved algorithm with the traditional single-objective algorithm PSO and QPSO and the second experiment is to compare our improved algorithm with the classical multi-objective algorithm NSGA-II.

The simulation parameters were as follows. The total number of physical servers was 100; the number of VMs was 200. All servers had the same allocation of resources: CPU with ten processing units, 20 GB of memory, and 100 M bandwidth. 200 VMs were equally divided into four types. The 200 requests for VMs arrived after optimization, and requests arrived in the same order for different configurations of the algorithm. For the optimization methods, the maximum size was 50, the maximum number of iterations was 500, and w_1 and w_2 were -0.5, if the current resource use of a physical server decreased by 10 %, or if there was a heat migration. The parameters of the QPSO are set as: delta = 0.001* p_i, the aberrance probability $P_m = 0.5$.

4.1 Experiment I

We verified the performance of our improved method by comparing it with two other single-objective PSO algorithms PSO and QPSO. The first was optimized for resource use, and the second was optimized for migration times. In our improved method, we set the investigated proportion of resource use to 0.6 and the investigated proportion of VM migration times to 0.4. The results of these three methods are shown in Fig. 4.

The results in Fig. 4 demonstrate that our proposed improved method for cloud computing resource allocation

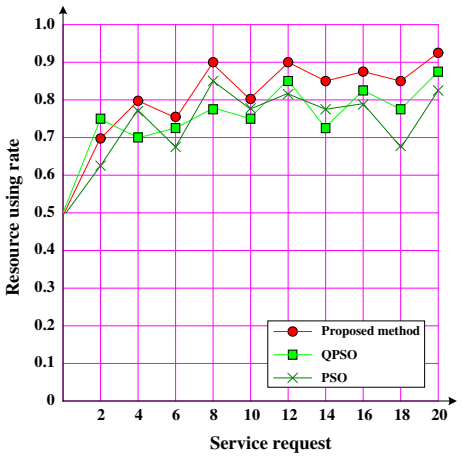

Fig. 4 Comparative experiments of recourse using the rate of three methods

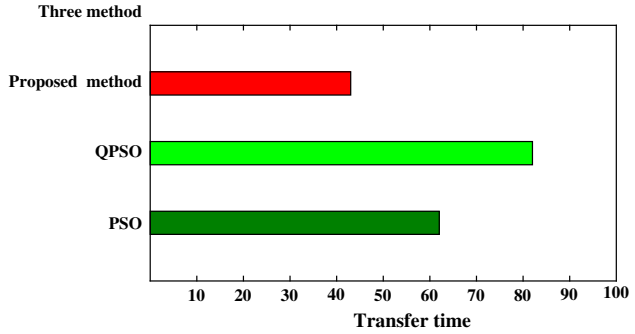

Fig. 5 Comparative experiments of migration times of the three methods

of VMs is very efficient. During the virtual machine deployment process and after stabilization, the resource use is significantly higher than the PSO and QPSO. We also compared the migration times of the three methods, as shown in Fig. 5.

The results in Fig. 5 demonstrate that the proposed approach has obvious advantages in terms of migration times. For the same population size, its migration frequency is 43 and the two other PSO techniques, PSO and QPSO, are 82 and 63. Finally, we investigated the execution time of three optimization methods from 100 iterations to 500 iterations. These experimental results are shown in Table 1.

The results in Table 1 demonstrate that when the number of iterations increases, the execution times of the three methods increase. We found that the execution time gap between QPSO and our method is very small; however, the execution time gap between PSO and our method or QPSO is very big. This is because the traditional PSO algorithm model is easy to fall into local optimum. QPSO greatly enhanced the efficiency of the search and can compensate for the lack of PSO. It has a wide research foreground and our method can effectively reduce premature convergence and greatly increase the probability of obtaining the global optimum. It is interesting that the results of our proposed algorithm suggest that the trend is linear, and not exponential.

The first experiment shows that on comparison of our improved method with the traditional single-objective algorithms, PSO and QPSO, our method was feasible and efficient.

4.2 Experiment II

To demonstrate the effectiveness of the proposed method, we designed experiments to compare NSGA-II (Deb et al. 2000) and our method IMOPSO. We define that f_1 represents resource utilization, f_2 represents migration time, and f_3 represents execution time. Statistical comparison data for the two algorithms are shown in Table 2. The convergence curves of the algorithms are shown in Figs. 6, 7, 8 and 9. Figure 6 shows the Pareto optimal solution set distribution of NSGA-II after running 100 generations after an experiment. Figure 7 shows the Pareto optimal solution set distribution of IMOPSO after running 100 generations after an experiment. Figure 8 shows the Pareto optimal solution set distribution of NSGA-II after running 500 generations after an experiment. Figure 9 shows the Pareto optimal solution set distribution of IMOPSO after running 500 generations after an experiment.

As can be seen in Figs. 6 and 7, the obtained Pareto optimal solution set has better convergence and distribution than that from NSGA-II. As seen from Figs. 8 and 9, the obtained Pareto optimal solution set can quickly converge to the Pareto front and the possible solutions are closer to the optimal solution than that from NSGA-II. This result was due to the improved approach introduced into the IMOPSO, which offered an effective combination of global and local searches. The global search is deductive for the evolution moving toward the Pareto front, while the local search can be used to explore more feasible solutions in unknown regions. From Table 2, the results of the IMOPSO algorithm are better than NSGA-II and the advantages of our method are more obvious; the experimental results show that the IMOPSO search performs well because it maintains good population diversity.

Table 1 Execution times of three optimization methods (statistical comparison 100 times)

Optimization methods	Comparison results					
	Iterations	100	200	300	400	500
Proposed method	Time (s)	5.313	6.933	10.112	14.521	20.598
QPSO	Time (s)	5.407	7.009	11.353	15.277	21.832
PSO	Time (s)	12.201	14.127	17.029	19.445	25.026

Table 2 Pareto optimal solution set comparisons of the two algorithms (statistical comparison 100 times)

Iterations	Algorithm	f_1 (resource utilization)	f_2 (migration time) (s)	f_3 (execution time) (s)
100	IMOPSO	0.36	0.367	12.01
	NSGA-II	0.33	0.660	13.55
500	IMOPSO	0.35	0.402	13.10
	NSGA-II	0.34	0.941	13.49

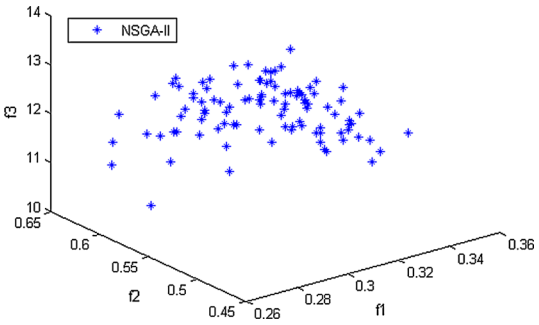

Fig. 6 Pareto optimal solution set distribution of NSGA-II (100 generations)

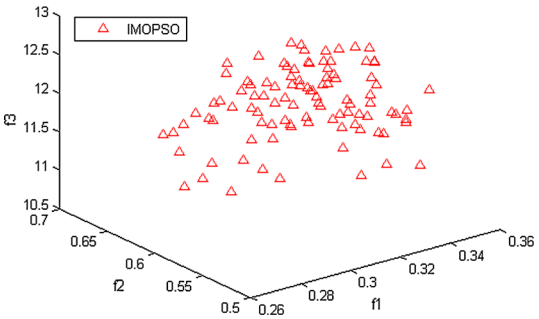

Fig. 7 Pareto optimal solution set distribution of IMOPSO (100 generations)

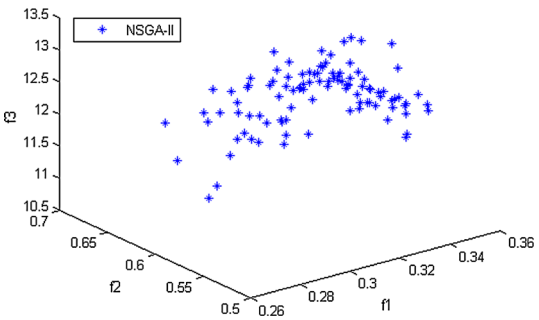

Fig. 8 Pareto optimal solution set distribution of NSGA-II (500 generations)

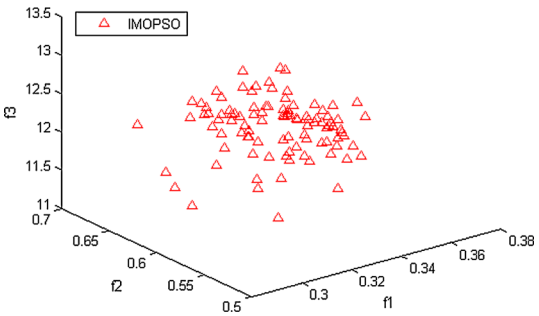

Fig. 9 Pareto optimal solution set distribution of IMOPSO (500 generations)

This experiment shows that IMOPSO can search effectively, maintain the population diversity, and quickly converge to the Pareto optimal solution without losing stability. The obtained Pareto optimal solution set has a better convergence and distribution than the comparative method.

5 Conclusion

In view of the deployment of virtual machine (VM) in cloud computing, considering the VM resources deployment in cloud computing environment and with the advantage of PSO, we propose the multi-objective comprehensive evaluation model for the dynamic deployment of virtual machines in this paper, the model based on some key factors including resource use and migration times, and then use an improved multi-objective particle swarm optimization (IMOPSO) to solve the problem. We have designed two simulation experiments: the first experiment was to compare our improved method with the traditional single-objective algorithms PSO and QPSO and our method was feasible and efficient; the second experiment was to comparen our improved algorithm with the classical multi-objective algorithm NSGA-II. The first experimental results show that on comparison of our improved algorithm with the traditional single-objective algorithms PSO and QPSO, our method is feasible and efficient. The second experimental results show that it maintains good population diversity and can quickly converge to the Pareto front. The obtained Pareto optimal solution set has a better convergence and distribution than a comparative method.

Dynamic deployment of virtual machines is a key issue of cloud computing; there are still many areas to be studied, especially for dynamic environments. We face not only the complexity of the resources, but also the actual application process including not only the combination of resources and task allocation, but also other follow-up steps. Of course, this is the next step in this research: a future work.

Acknowledgments This research is supported by the National Natural Science Foundation of China (Grant No. 61272382), Science and Technology Foundation for the Universities of Guangxi Province (Grant No. 2013ZD060), the Hunan Provincial Natural Science Foundation of China (Grant No. 12JJ6063), and Guangdong Province Science and Technology Project (Grant No. 2012B010100037).

References

Aksaç A, Ozturk O, Ozyer T (2011) Real-time multi-objective hand posture/gesture recognition by using distance classifiers and finite state machine for virtual mouse operations. In: IEEE 7th International Conference on electrical and electronics engineering (ELECO), pp 457–461

Armbrust M, Fox A, Griffith R et al (2010) A view of cloud computing. Commun ACM 53(2):50–58

Blackwell T, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans Evol Comput 10(2):459–472

Buyya R, Yeo CS, Venugopal S et al (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gen Computer Syst 25(4):599–616

Calheiros RN, Ranjan R, Beloglazov A et al (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Chaisiri S, Lee BS, Niyato D (2012) Optimization of resource provisioning cost in cloud computing. IEEE Trans Serv Comput 5(2):164–177

CloudSim (2009) a novel framework for modeling and simulation of cloud computing infrastructures and services. University of Melbourne, Melbourne

Coello Coello CA, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the IEEE Congress on evolutionary computation (CEC'02), vol 2, pp 1051–1056

Cruz C, González JR, Pelta DA (2011) Optimization in dynamic environments: a survey on problems, methods and measures. Soft Comput 15(7):1427–1448

Deb K, Agrawal S, Pratap A et al (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Lect Notes Computer Sci 1917:849–858

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on micro machine and human science (MHS'95). IEEE, pp 39–43

Foster I, Zhao Y, Raicu I et al (2008) Cloud computing and grid computing 360-degree compared. In: IEEE grid computing environments Workshop (GCE'08), pp 1–10

Heo JS, Lee KY, Garduno-Ramirez R (2006) Multiobjective control of power plants using particle swarm optimization techniques. IEEE Trans Energy Convers 21(2):552–561

Hyser C, Mckee B, Gardner R, Watson BJ (2007) Autonomic virtual machine placement in the data center. HP Labs Technical Report

Janson S, Merkle D, Middendorf M (2008) Molecular docking with multi-objective particle swarm optimization. Appl Soft Comput 8(1):666–675

Kong X, Lin C, Jiang Y et al (2011) Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. J Netw Computer Appl 34(2):1068–1077

Kourai K, Chiba S (2011) Fast software rejuvenation of virtual machine monitors. IEEE Trans Depend Secure Comput 8(4):839–851

Maguluri ST, Srikant R, Ying L (2012) Stochastic models of load balancing and scheduling in cloud computing clusters. In: IEEE (2012) Proceedings of INFOCOM. IEEE, pp 702–710

Mahdavi I, Aalaei A, Paydar M-M, Solimanpur M (2011) Multi-objective cell formation and production planning in dynamic virtual cellular manufacturing systems. Int J Prod Res 49(21):6517–6537

Maurer M, Emeakaroha VC, Brandic I, Altmann J (2012) Cost-benefit analysis of an SLA mapping approach for defining standardized Cloud computing goods. Future Gen Computer Syst 28(1):39–47

Mohammadi E, Karimi M, Saeed RH (2011) A novel virtual machine placement in cloud computing. Aust J Basic Appl Sci 5(10):1549–1555

Nguyen Q-H, Nien PD, Nam N-H, Nguyen H-T, Nam T (2013) A genetic algorithm for power-aware virtual machine allocation in private cloud. In: Lecture notes in computer science, v7804-LNCS, pp 183–191

Parsopoulos KE, Vrahatis, MN (2002) Particle swarm optimization method in multi-objective problems. In: Proceedings of the ACM 2002 Symposium on applied computing (SAC'2002), pp 603–607

Reyes-Sierra M, Coello Coello AC (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int J Comput Intell Res 2(1):287–308

Sato K, Samejima M, Komoda N (2013) Dynamic optimization of virtual machine placement by resource usage prediction. In: 2013 11th IEEE International Conference on industrial informatics (INDIN). IEEE, pp 86–91

Sindelar M, Sitaraman RK, Shenoy P (2011) Sharing-aware algorithms for virtual machine colocation. In: Proceedings of the 23rd ACM symposium on parallelism in algorithms and architectures. ACM, pp 367–378

Sindhu S, Mukherjee S (2011) Efficient task scheduling algorithms for cloud computing environment. Commun Computer Inf Sc 169(1):79–83

Stillwell M, Schanzenbach D, Vivien F et al (2010) Resource allocation algorithms for virtualized service hosting platforms. J Parallel Distrib Comput 70(9):962–974

Takahashi S, Nakada H, Takefusa A et al (2012) Virtual Machine packing algorithms for lower power consumption. In: 2012 IEEE 4th International Conference on cloud computing technology and science (CloudCom). IEEE, pp 161–168

Warneke D, Kao O (2011) Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. IEEE Trans Parallel Distrib Syst 22(4):1045–9219

Wilcox D, McNabb A, Seppi K (2011) Solving virtual machine packing with a reordering grouping genetic algorithm. IEEE Congr Evol Comput (CEC) 2011:362–369