Cluster Comput
DOI 10.1007/s10586-015-0447-7

Immune optimization of task scheduling on multidimensional QoS constraints

Hejun Jiao[1,2] · Jing Zhang[1] · JunHuai Li[1] · Jinfa Shi[3] · Jian Li[2]

Received: 6 July 2014 / Revised: 17 November 2014 / Accepted: 13 March 2015

Abstract Aiming at the sensitive issues of service quality in cloud computing, a task scheduling tactic with multidimensional QoS constraints is studied. Based on cluster service and user QoS preference, this article constructs an immune optimization model to make a description through formulas and quantify the performance constraints; the utility function of multidimensional QoS is given and then the immune optimization operation is performed with the antibodies. It is beneficial to increase the prediction accuracy of the equality evaluation, and the search for a Pareto optimal set of multiobjective optimization problems is implemented. Finally, the optimum node distribution structure with the highest utility value is obtained. It's shown that the approach gives sufficient consideration of multidimensional user QoS requirements. The results from the test show a significant improvement in average rate of equipment utilization, service time and response time compared to similar algorithms.

Keywords Cloud computing · Multiple QoS parameter constraint · Immune optimization · Application preference · Task scheduling

✉ Hejun Jiao
david.710@163.com

Jing Zhang
ZhangJing@xaut.edu.cn

Jinfa Shi
jfshi@zzia.edu.cn

[1] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

[2] Department of Computer Science & Engineering, Henan Institute of Engineering, Zhengzhou 451191, China

[3] School of Management Science and Engineering, Zhengzhou Institute of Aeronautical Industry Management, Zhengzhou 450015, China

1 Introduction

Developed from grid computing, distributed computing and utility computing technology, cloud computing is to blend Web2.0 and system virtualization and its data processing can be moved from the PC or server to the Internet in the cluster by using the high-speed Internet. Task scheduling is a common problem in cloud computing. An efficient scheduling tactic can take full advantage of cloud processing ability to improve the application, and it is particularly important for cloud rescheduling research. Because the tasks are flexible in cloud computing, we should try to improve the throughput of the system resource and the optimal span, and we also should consider security, load balancing and user satisfaction in a massive task for the scheduling. Scheduling algorithms generally concern just a single service quality (QoS, the quality in the service) constraint, for example, Li in [1] proposed a genetic algorithm of the task scheduling based on double fitness. He only considered the completion time, unable to meet the needs of multidimensional QoS constraints.

A key issue that must be addressed is to improve the task scheduling algorithm and ensure the quality of service during the virtual machine scheduling in cluster. The cloud computing platform will create virtual machines according to the QoS constraints after receiving task orders and perform the corresponding user requests, such as the open Eucalyptus platform. Under the multidimensional QoS constraints, however, request may conflict with each other, which is likely to increase the difficulty of the scheduling problem. We must fully consider QoS performance, various benefits and the degree of load balancing about resource nodes in the task scheduling design; the immune clone algorithm can solve the task scheduling problems for multiple QoS constrained in cluster. This article analyzes the massive task processing

Published online: 24 March 2015

characteristics in cloud computing, such as difficult deployment, low reliability and performance.

This article also points out the necessity of encoding improvement. And then a concept of QoS preference is proposed to convert fitness function of the scheduling problem into objective function, and finally, we do experiments on Eucalyptus platform, making the algorithm conform to the actual application, at the same time strengthening the robustness of the algorithm by using the thought of clonal selection algorithm in the iterations for inference.

2 Related work and problem analysis

The goal of this work is to provide an optimal task scheduling algorithm for the virtual machines in cloud computing. The virtual machines are typically large-grained. When users deploy virtual resources, they submit virtual resource request to the elastic cloud platform. The cloud platform control center accepts the user's request, and starts the sampling mechanism in the model to find a host machine that meets virtual machine creation. We call the tasks within a schedule meta-task; the meta-tasks are independent from each other and corresponding to a single service in set. A service can be measured by the QoS. Finally the utility function in multiple QoS constrained model could transfer constraint into utility value known as the user satisfaction. For example, Polo in [2] presented an application-centric task scheduler for Hadoop (Apache's open source implementation of a MapReduce framework). Its scheduler is able to calculate the estimated completion time for each MapReduce job in the system.

Gogulan in [3] put forward an improved ant colony algorithm about cloud scheduling under the constraint of time, cost and reliability. The algorithm only makes a simple comparison under Cloudsim. It does not consider time limit and load balancing. Li in [4] put forward an optimized Chord scheduling algorithm under the restriction of utility and efficiency. This algorithm could only consider two QoS constraints, utility and efficiency, and did not consider network and storage; the efficiency is not high. Ye in [5] put forward a kind of load balance mechanism based on QoS, and established a mapping between the measurement of resources and the QoS properties. There was a lot of human intervention when using the mapping, the method was too complex, and did not give detailed of the description. Yu, put forward an improved genetic algorithm based on chromosome coding and fitness function, which had great reference value for the scheduling.

This article defines the four most common QoS to describe a service. The service s_q that the task t_p suggests can be expressed as $s_q\left(\alpha_{pq}, \beta_{pq}, \gamma_{pq}, \theta_{pq}\right)$ [6]. The four parameters in brackets are respectively the time spent, expense, priority and load that t_p uses the service s_q. Each task corresponds to one or more service node.

Definition 1 V is the collection of computing resources $V = \{v_1, v_2, v_3, \ldots, v_m\}$, and it represents a group of heterogeneous resources in cloud.

Definition 2 For a task set with n tasks $T\{t_1, t_2, t_3 \ldots t_n\}$ each task t_i consists of d_i service nodes which form QoS constraint space S_i, s_j represents the jth $(1 \leq j \leq d_i)$ finite set of QoS constraint about t_i $s_j = \{s_1, s_2, \ldots, s_{di}\}$ represents the d_ith dimension of task t_i, $s_{d_i} = \left\{\alpha_{id_i}, \beta_{id_i}, \gamma_{id_i}, \theta_{id_i}\right\}$ is a point on the QoS space.

$$
\begin{aligned}
&f(S_1, S_2, \ldots, S_n) = \left(\sum_{i=1}^{n} \min\left(\alpha_{ij}\right),\right. \\
&\left.\times \sum_{i=1}^{n} \min\left(\beta_{ij}\right), \max\left(\gamma_{ij}\right), \sum_{i=1}^{n} \min\left(\theta_{ij}\right)\right), \\
&\quad s.t.\ \forall i \exists j,\ 1 \leq j \leq d_i, \\
&\sum_{i=1}^{n} \min\left(\alpha_{ij}\right) \leq T, \sum_{i=1}^{n} \min\left(\beta_{ij}\right) \leq C, \min\left(\sum_{i=1}^{n} \theta_{ij}\right) \leq L
\end{aligned}
\tag{1}
$$

Among them, T, C and L are time, cost and the load QoS constraints of the user respectively. It has been proven that the utility function is difficult to solve and has an optimal solution:

Proof We make the assumption that the question contains n tasks and m service nodes; the number of tasks is greater than the service node. Given the principle of average distribution,we adopt the list method. So the minimum complexity is $O((n/m)!)$ the more tasks we have, the more difficult it is to solve. END. □

Theorem 2 *If the four constraint parameters have characteristics of relative independence and there is an optimal solution in the solution space. For any task, then, the optimal solution exists in the service list. The time, cost, and priority and load are optimal.*

Proof Suppose the optimal solution is

$$
\begin{aligned}
&f^*(S_1, S_2, \ldots, S_n) \\
&= \left(\sum_{i=1}^{n} \min\left(\alpha_{ij}\right), \sum_{i=1}^{n} \min\left(\beta_{ij}\right), \max\left(\gamma_{ij}\right), \sum_{i=1}^{n} \min\left(\theta_{ij}\right)\right)
\end{aligned}
$$

□

If there is a selected service task corresponding to t_i, and if the time or the priority of the service is not optimal, there must be service $s_j\left(\alpha_{aj}, \beta_{aj}, \gamma_{aj}, \theta_{aj}\right)$ for the task t_i, $\alpha_{ai} > \alpha_{aj}$ or

$\lambda_{ai} < \lambda_{aj}$, then we replace s_i with s_j in the optimal solution, which result in the undesirable total time or total priority, obviously contradicting the premise above, end [7].

The necessary condition of theorem 2 is also true. Suppose there is an optimal service in solution space. Conflicts can arise if there is no optimal solution. Its proof can be omitted.

It can be concluded that we obtain the optimal solution, as long as we find the optimal service node of n tasks in turn. But in cloud computing, multiple QoS attributes influence each other, which conflicts with theorem two on conditions. Therefore, there is no optimal solution for scheduling. From theorem 1, using the list method to solve the problem is not feasible; therefore, problem U needs an evolutionary algorithm to achieve approximate solution.

3 Immune optimization of scheduling with multiple QoS preference constraints

Using a bit of antibodies to identify their larger antigen is the evolution goal of immune system. This article uses the immune genetic algorithm to solve the problem. The main operation includes immune selection, immune recognition and antibody operation.

Now that we have formally given a definition of the utility function for QoS constraints, we can convert each constraint to utility. The total utility is the sum of each utility; the total utility function is defined as (2):

$$Util\,(qs) = \max \sum_{i=1}^{n} \sum_{u=1}^{us} \sum_{h=1}^{k} w_{iu}\,(h)\, Util_{iu}\,(qs_{iu}\,(h)) \times \gamma_{ih}$$
$$s.t.\ \ 0 \le w_{iu}(h) \le 1; \sum_{h=0}^{k} w(h) = 1 \tag{2}$$

In the formula, $Util_{iu}\,(qs_{iu}\,(h))$ is the utility value of the hth QoS $qs_{iu}\,(h)$ corresponding to the task t_i, us is the number of users, $w_{iu}(h)$ is the weight corresponding to the hth QoS. The goal of task scheduling is the total utility maximization. The algorithm is embedded with computation of the utility. The specified parameters are set in Sect. 4 of this paper.

3.1 Immune recognition

The chromosome of antibody can be generated by symbol code. Each antibody is a scheduling scheme and can be defined as a data structure in the program. The genetic value in Chromosomes represents the assigned resource number [8]. The parameters and values are shown in the following Table 1 with a size of N. The antibody is generated based on antigen, which usually refers to the objective function. The initial solution of the first-generation accords with the formula 3 for IC_j, which represents the jth antibody of task set by user 1 in task set. The first generation of antibody population satisfies the formula 4.

$$IC_j\,(1) = \left(ic_j^1, ic_j^2, \ldots, ic_j^k, \ldots, ic_j^{tn}\right), ic_j^k \in \{1, 2, \ldots, m\} \tag{3}$$

$$IC\,(1) = (IC_1\,(1), IC_2\,(1), \ldots, IC_z\,(1), \ldots, IC_N\,(1)), \times z \in \{1, 2, \ldots, N\} \tag{4}$$

In the formula, ic_j^k is the serial number of the physical resources assigned by the kth task ts_k of the jth antibody, tn represents the total number of tasks for all users, which satisfied formula 5. The memory is empty in the initial antibody population. For multi constraints' cases, it is necessary to select a random algorithm to generate an antibody.

$$tn = \sum_{i=0}^{us} ts(i), i \in \{1, 2, \ldots, us\} \tag{5}$$

In the formula, $ts(i)$ is the number of cloud requests and us is the number of users.

The affinity QC_vQC_w between the antibody IC_v and IC_w satisfies the formula 6, the smaller the similarity, the lower affinity. Diversity among antibodies is encouraged. When the $QC_vQC_w = 1$, the two incidents are completely different.

$$QC_vQC_w = \frac{1}{1 + H(N)} = \frac{1}{1 + \frac{1}{M}\sum_{j=1}^{M} H_j(N)} = \frac{1}{1 + \frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N} -p_{ij}\log_2 p_{ij}} \,|N = 2 \tag{6}$$

In the formula, p_{ij} is the probability of jth character x for N antibodies, $i \in \{1, 2, \ldots, N\}, j \in \{1, 2, \ldots, M\}$. M is the length of each antibody gene; N is the scale number of antibody.

The affinity QC_vQY between the antibody IC_v and its antigen satisfies the formula 7, which is used to indicate a degree that the antibody recognizes the antigen. The greater the affinity, the more utility the antibody IC_v brings about.

$$QC_vQY = \sum_{i=1}^{m} \sum_{h=1}^{k} w_{ic_v^i u}(h) Util_{ic_v^i u}\left(qs_{ic_v^i u}(h)\right) \times \gamma_{ic_v^i h}$$
$$s.t.\ \ 1 \le v \le N; 1 \le u \le us \tag{7}$$

3.2 Antibody operation

Antibody operation mainly includes immune replication, immune cross and selection. This article chooses the adaptive

Table 1 Immune algorithm parameters

Parameter	Description	Value
$H_j(N)$	The jth information entropy of N antibodies	The sum of $-p_{ij}\log_2 p_{ij}$ $1 \le i \le N$
It_{max}	The maximum number of iteration	$c{\cdot}\log_2(space)$
p_c	Immune replication rate	0.135
p_r	The immune cross probability	0.9
p_m	Predefined mutation probability	0.009
ψ	Random number of Gauss	0–1
is	The current antibody generation	/
λ	Similarity constant	$0.9 \le \lambda \le 1$

multi-objective optimization. The replication operation FC^C in the antibody group IC is as shown in formula 8:

$$\begin{aligned} IC^*(is) &= FC^C(IC(is)) \\ &= \Big(FC^C(IC_1(is)), FC^C(IC_2(is)), \ldots, \\ &\quad \times FC^C(IC_N(is))\Big) \end{aligned} \tag{8}$$

In the formula, $IC_i^*(is) = FC^C(IC_i(is)) = L_i \times (IC_i(is))^T$, L_i is composed of l_i dimensional row vectors of 0 or 1. If the element is 1, then it is the copy of l_i to $IC_i(is)$.

$$\begin{aligned} l_i(is) &= y(N^C, QC_iQY(IC_i(is)), p_c) \\ &= Int\left(N^C \bullet \frac{QC_iQY(IC_i(is))}{\sum_{i=1}^{N} QC_iQY(IC_i(is))} \bullet p_c\right) \end{aligned} \tag{9}$$

In the formula, $N \le N^C$. N^C is related with the size of population. $Int(\bullet)$ rounds up the value, the size of the replication operation is adjusted adaptively based on the affinity QC_vQY and the ratio of immune replication. When the affinity between antibody and antigen is sizable, it needs a larger replication scale. Of course, the population size cannot be too huge, which will affect the efficiency of execution. In practice, we will copy the larger utility value in the whole population. After replication, the populations become:

$$IC^*(is) = (IC_1^*(is), IC_2^*(is), \ldots, IC_{N^*}^*(is)), N^* = \sum_{i=1}^{N} l_i \tag{10}$$

Satisfy formula 11,

$$\begin{aligned} IC_j^*(is) &= \left(IC_{ij}^*(is)\right) = \left(IC_{i1}(is), IC_{i1}(is), IC_{il_i}(is)\right) \\ IC_{ij}^*(is) &= IC_{ij}(is) = IC_i(is) \end{aligned} \tag{11}$$

Through the genetic operations such as the cross and mutation of individuals, we continually produce a fresh generation of individuals, and we choose the optimal antibody from the antibody population and memory $IC_{history}$. If the best individual is in the contemporary population IC_{random}, then replace nothing, or replace the worst individual, leading to the formation of new species, the selection operation of the antibody group is shown in formula 12. For convenience, we can fix the number of groups that need to be copied.

$$\begin{aligned} IC^*(is) &= FC^S(IC(is)) = FC^S(IC_1(is), \\ &\quad IC_2(is) \ldots, IC_{history+random}(is)) \\ &= \left(IC_1^*(is), IC_1^*(is), \ldots, IC_N^*(is)\right) \end{aligned} \tag{12}$$

The cross strategy applies to all partial matches. We select two antibodies with the highest contemporary reproduction rate and the best antibody concentration to cross. The formula EV_i that denotes selectivity at reproduction is as follows:

$$EV_i = \frac{QC_iQY\prod_{j=1}^{N}(1 - AS_{ij})}{CV_i\sum_{j=1}^{N} QC_jQY} \tag{13}$$

$$AS_{ij} = \begin{cases} QC_iQC_j & QC_iQC_j \ge \vartheta \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

ϑ and φ are the pre-determined threshold. The antibodies with low expectations are restrained at reproduction. CV_i represents the density, among them, $CV_i = \frac{1}{N}\sum_{j=1}^{N} AC_{ij}$,

$$AC_{ij} = \begin{cases} 1 & QC_iQC_j \ge \varphi \\ 0 & \text{otherwise} \end{cases}$$

3.3 Immune mutation

The definition of immune mutation is as follows:

$$\begin{aligned} ic_i^k(is+1) &= ic_i^k(is) + \varepsilon_k \times \left(ic_{best}^k(is) - ic_i^k(is)\right), \\ &\quad i \in \{1, 2, \ldots, N\} \end{aligned} \tag{15}$$

$ic^k_{best}(is)$ is the kth place value of the gene for the best individual in the current iteration. ε_k is the variable coefficient. It satisfies the following formula.

$$\varepsilon_k = \psi \bullet (1 - C) \bullet \left(1 + sgn(\phi - C) \bullet \frac{f(IC_i(is))}{f(IC_{best}(is))}\right) \tag{16}$$

Among them, ψ is the random number of Gaussian distribution. Its value is between 0 and 1. ϕ is concentration threshold of the optimal antibody (the initial value was 0.80), and C is the concentration of optimal antibody. It is set to:

$$C = \frac{\sum \frac{f(IC_i(is))}{f(IC_j(is))}}{N}, \varsigma \leq \frac{f(IC_i(is))}{f(IC_j(is))} \leq \frac{1}{\varsigma}; 1 \leq j \leq N \tag{17}$$

Among them, the adjustable parameter $\varsigma \in (0, 1)$, the initial value is 0.95. The concept of 'concentration' refers to the antibody ratio, which means that the level of fitness similarity between the ith and other antibodies is less than ς in the mixed population. The objective is to suppress the high concentration antibody and encourage low concentration antibody in the choice of them.

3.4 Algorithm description

The immune algorithm is proposed in this paper to seek the best mapping between tasks and resources by using the population, to suppress the antibodies with low reproduction rate, cross individuals with high replication rate and mutate some superior individuals, considering characteristics of cloud resource scheduling with application preferences and multi-objective QoS optimization. We use immune selection, immune replication and cross, immune selection and mutation operation to copy the excellent individual, and eliminate the worst individuals. The pseudo code is given by Fig. 1, we can schedule tasks dynamically according to the situation.

The time complexity of the algorithm includes immune selection, immune cross, immune replication and mutation. The default size of immune antibody is N; the size is N^* after immune replication. In an immune operation, the time complexity of the first immune selection is $O(2*M*N)$; the time complexity of immune replication is $O(M*N^*)$; the time complexity of immune cross is $O(M^2*N^*)$; the time complexity of the second immune selection is $O(2*M*N^*)$; the time complexity of immune mutation is $O(2*N^*)$.

4 This algorithm analysis and comparison

This paper focuses on the change process of the best individual fitness and average fitness in population during the iterations [9]; the test data is moderate, the number of tasks labeled n is 100, where each virtual machine VCPU number is between 1 and 2. There are four factors to affect the performance of scheduling, the time spent, expense, priority and load. The user preferences are set to (1/4, 1/4, 1/4, 1/4). the task computation is between 10,000 and 50,000 Bytes; the data is transmitted via the 100 Mbps switch; the priority is divided into four simple levels (1, 2, 3, 4), the larger the value, the higher the priority. The valid execution time of task t_i satisfies formula 18.

$$vt_i = \psi \bullet \left(\frac{len_i}{1.05 \bullet peNumber} + \frac{len_i}{bw}, \frac{len_i}{0.95 \bullet peNumber} + \frac{len_i}{bw}\right) \tag{18}$$

In the formula, ψ is the random number of Gaussian distribution. Its value is between 0 and 1. len_i is the task computation. bw is the network bandwidth. Its value is 10 MBps. $peNumber$ represents each virtual machine's processing power. To avoid uneven loads, we use load to measure the usage degree of cloud resources, as shown in formula 19. To increase the operation speed and scheduling precision for time and cost, the normalized data is mapped to [0, 1] interval, as shown in formula 20. in the formula, x'_{ij} is the utility index of resource m_j in the the ith dimension. $curx_{ij}$ represents the current utility value.

$$load_i = \frac{len_j}{\sum_{j=1}^{m} len_j} / \frac{vt_i}{\sum_{i=1}^{n} vt_i} \tag{19}$$

$$x'_{ij} = \left(curx_{ij} - x_{\min}\right) / \left(x_{\max} - x_{\min}\right) \tag{20}$$

Figure 2 shows one result of the algorithm and genetic algorithm; we can see from the figure that the utility value span of the proposed algorithm is greater than the average fitness of genetic algorithm, which shows clearly that multiple QoS immune algorithm is not easy to fall into local optimum. However, for the genetic algorithm, the average utility value of population continues to rise with the increasing of iterations, which shows that multiple clones of an individual with higher utility value affect the fulfillment of the optimal solution. The genetic algorithm mimics Darwinian natural selection, where "fitness" selects individuals for survival, breeding, and, hence, adaptive mutation. Table 2 shows the configuration parameters for the data center.

The comparison of the best individual utility value and average utility value between the multi-dimensional QoS immune algorithm and AFSA is shown in Fig. 3 [10]. Although the average utility value and the best individual fitness of the proposed algorithm are less volatile than AFSA, after a certain number of iterations, the algorithm has a great

Fig. 1 Multidimensional QoS immune algorithm

1:Initialize Cross Rate p_c, Mutation Rate p_m, ψ ,antibody density C, It_{max}, λ, and initialize antibody population $IC(N)$ by class Genetic
2:while $is < It_{max}$ do
3: initialize preference vector $w_{iu}(h)$
4: for all tasks in meta-task T, $t_i \in T$
5: for all antibodies in $IC(is)$, $ic_j^k \in IC_i(is)$
6: Compute $Util(qs)$
7: end for
8: end for
9: perform immune select by (12) for $IC(is)$
10: copy $IC_i(is)$ according to $IC_{history}$, IC_{random} and (10), get new antibody population $IC^*(is)$
11: compute affinity QC_vQC_w and QC_vQY through (6) and (7)
12: perform immune cross through (13) and (14) , get new antibody population $IC^{**}(is)$
13: perform immune reselect by (12) for $IC^{**}(is)$
14: set the Gaussian random number ψ and density threshold ϕ
15: perform immune mutation by (15) , and get new antibody population $IC^{***}(is)$
16: is+=1
17: reduce population sizes to $IC(is)$
18: output the optimal individual
19: end while
20:end

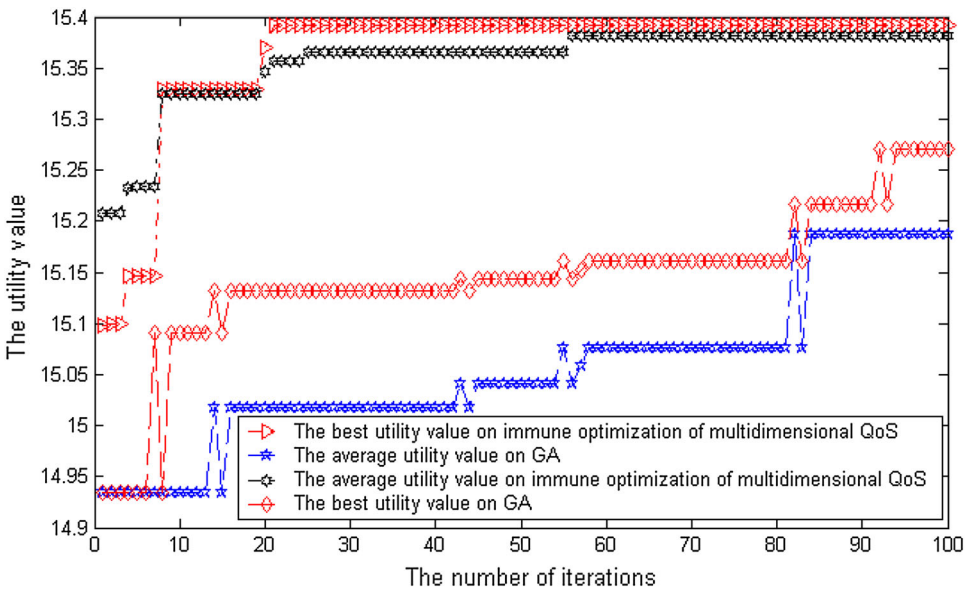


Fig. 2 The comparison of the best and average utility value between the proposed algorithm and GA

Table 2 Data center parameters

Cloud resource category	CPU number /amount	Memory/M	Cost ($/MI)	Appropriate task number	Virtual machines
Virtual Machine Class A	1	512–1024	5	[5,29]	4
Virtual Machine Class B	2	1024–2048	12	(5,20]	1
Virtual Machine Class C	2	1024	10	(5,25]	1
Virtual Machine Class D	1	1024–2048	8	[5,29]	4

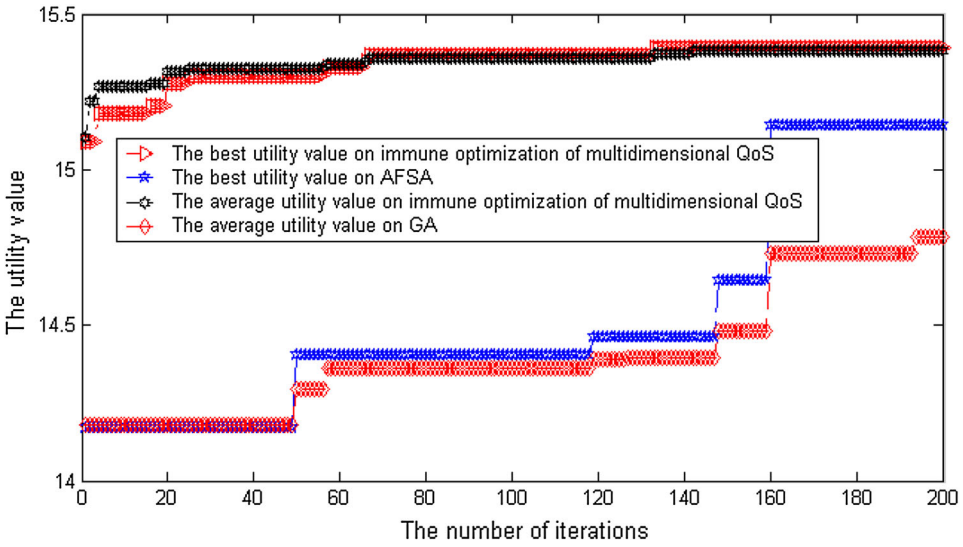


Fig. 3 The comparison of the best and average utility value between the proposed algorithm and AFSA

Table 3 The resource list of node controller

Physical server	CPU	Memory
Fronted F1	4	2
Node N1–N3, N6	2	2
Node N4–N5	2	4

advantage in searching for the optimal value; the effect of the proposed algorithm on the changes in the population is significant. Artificial fish swarm algorithm (AFSA) is a stochastic global optimization technique proposed lately.

It can be seen from Figs. 2 and 3 that the immune optimization of multidimensional QoS can converge rapidly and search out the global optima more frequently compared with the other two algorithms.

5 Experiment and performance analysis

Using the CentOS6 Linux operating system and Eucalyptus, we build six nodes and a front node. Then we compare the performance of physical server after tasks are allocated in this paper, the version of open Eucalyptus is 3.2.2. Resources are shown in Table 3. We should keep time synchronized during deployment, apply and manage virtual resources after node register. In order to evaluate the proposed multi-dimensional QoS immune optimization algorithm in cloud resource scheduling, we make a comparison between genetic algorithm and AFSA. The performance indexes of test analysis include the equipment utilization rate, the average response time and service time. Eucalyptus is an open source software for building Amazon Web Services-compatible private and hybrid clouds.Eucalyptus is broken into five components: Cloud Controller, Walrus,Cluster Controller, Storage Controller, Node Controller. These components are software services and are arranged in three layers: cloud, cluster, and nodes.

The equipment utilization rate of the task could be used as the system-level criterion of the scheduling algorithm. The scheduling algorithm which has the smallest average utilization rate can provide better service for the system. This paper also takes the service time and response time as a measure to evaluate the algorithm. The tasks now facing each Node are different, so are its three performance indexes. The experimental results are shown in Figs. 4, 5, and 6.

The pictures above show the performance comparison of four algorithms under same tasks and different nodes. The immune optimization of multidimensional QoS is better than other algorithms, especially in the response time; the GA algorithm is poorer in the service time. This is because the GA algorithm traps into local optima easily, the algorithm also has 200 iterations, but only deals with the narrow search within the scope of the population; the population size is not large. The improved AFSA increases the probability to

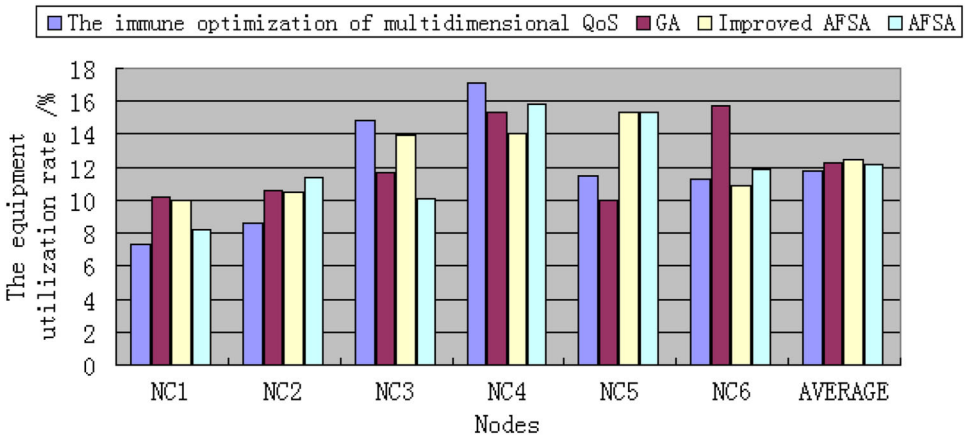


Fig. 4 The comparison of differences on equipment utilization rate

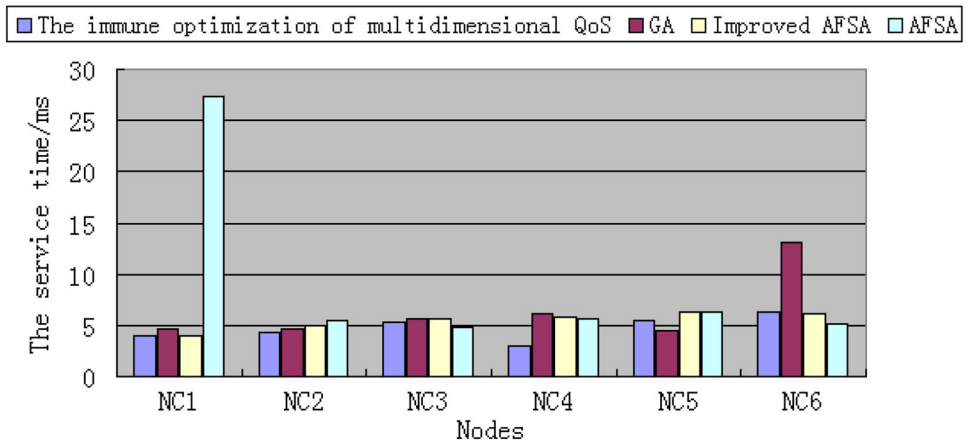


Fig. 5 The comparison of differences on service time

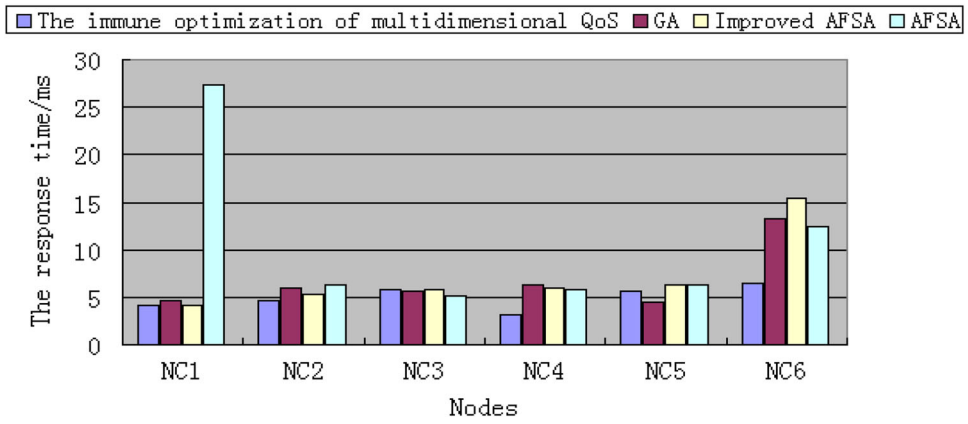


Fig. 6 The comparison of differences on response time

move to the optimum solution compared with the artificial fish algorithm. Relatively speaking, it enlarges the search scopes, and it reduces the effects of the results on the earlier stage on the late stage and improves the performance.

In conclusion, although the performance indexes are slightly better in each Node, they generally have superiority. The immune optimization of multidimensional QoS and the improved AFSA have better performance than GA and AFSA in the constraint model whether in the average equipment utilization rate or in the response time and service time, the immune optimization of multidimensional QoS uses the best performance, and this algorithm has less service time than GA, works much faster.

6 Summary and prospect

In this paper, we propose the immune optimization algorithm of multidimensional QoS and successfully achieve rapid multiobjective optimization by quantifying application

preferences and utility in multidimensional QoS space. The success rate of task scheduling is 100 %. Thus it can maximize the total utility value of user while completing the task allocation effectively. The analysis and experiments of the cloud platform show that the immune optimization algorithm is better than the GA,improved AFSA and general AFSA in the average equipment utilization rate, response time and service time, which can get good scheduling results.

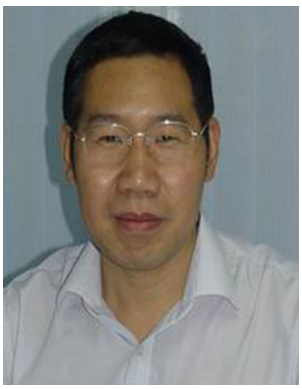
The next step is to consider preference QoS parameters more thoroughly, and dynamically adjust the virtual resource migration, verify the universality of the immune algorithm for the model with no QoS, study the theoretical model to the parameters of other intelligent algorithms and global search method in local performance.

Acknowledgments This work is partialy supported by the National Natural Science Foundation of China (Nos. 61172018, 71371172), Science and Technology Research Projects of Henan Province (No. 142102210037), Science and Technology Plan Projects on Water Conservation of Shaanxi Province (No. 2012-08), the Major Program of National Soft Science Research (No. 2013GXS2B010) and the Scientific Research Projects of Key laboratory of Shaanxi Province (No. 13JS084).

References

1. Li, J., Peng, J.: Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. J. Comput. Appl. **31**(1), 184–186 (2011)
2. Polo, J., Nadal, D., Carrera, D., Becerra, Y., Beltran, V., Torres, J., Ayguadé, E.: Adaptive task scheduling for multi-job MapReduce environments. XX Jornadas de Paralelismo (JP 2009), pp 96–101, A Coruña, pp. 16–18, 2009
3. Gogulan, R., Kavitha, A., Karthick Kumar, U.: An multiple pheromone algorithm for cloud scheduling with various QOS requirements. Int. J. Comput. Sci. Issues **9**(3), 232–238 (2012)
4. Li, B., Song, M., Song, J.: A distributed QoS-constraint task scheduling scheme in cloud computing environment: model and algorithm. Adv. Inf. Sci. Serv. Sci. **4**(5), 283–291 (2012)
5. Ye, F., Wang, Z., Zu, X., Wang, L., Zhang, X.: Research of a load balancing mechanism of cloud based on QoS. J. Chin. Comput. Syst. **33**(10), 2147–2152 (2012)
6. Wen, S., Chen, J., Guo, T.: Optimized virtual machine deployment mechanism in cloud platform. Comput. Eng. **38**(11), 17–19 (2012)
7. Zhao, J., Zeng, W., Liu, M., Zhang, X.: Grid computing workflow scheduling clonal selection algorithm with multi-QoS constraints. Pattern Recognit. Artif. Intell. **24**(5), 713–722 (2011)
8. Sun, W., Qin, Z., Li, M., Hu, J.: QIACO: an algorithm for grid task scheduling of multiple QoS dimensions. Acta Electron. Sin. **39**(5), 1115–1119 (2011)
9. Ben, F., Wang, Y.: Virtual machine resource allocation strategies based on fault-tolerant QoS in cloud computing. Microelectron. Comput. **30**(3), 136–139 (2013)
10. Neshat, M., Adeli, A., Sepidnam, G., Mehdi, T., Najaran, A.: A review of Artificial Fish Swarm Optimization methods and applications. Int. J. Smart Sens. Intell. Syst. **5**(1), 107–148 (2012)

Hejun Jiao received his B.Sc. and M.Sc. both from Henan University of Science and Technology in 2005 and 2008. He was born in Xinxiang city, Henan province, P. R. China in June, 1981. He has been working at Henan Institute of Engineering since 2008. Now he is a Ph.D. student in Xi'an University of Technology. His major research interests include cloud computing, computer network, pattern recognition etc.



Jing Zhang male, doctor, professor, doctoral supervisor. He was born in Baoji city, Shaanxi province, P. R. China in November, 1952. He received bachelor degree in Department of Automatic Control of Xi'an University of Technology in 1981, the master degree in Department of Software and Theory of Xi'an JiaoTong University in 1986, and the doctor degree in Department of Systems Engineering of Xi'an JiaoTong University in 1994. He has worked in Department of Computer of Xi'an University of Technology since 1977, and now is a professor and the Ph.D. supervisor of School of Computer Science and Engineering, Xi'an University of Technology.

JunHuai Li received the B.S. degree in electrical automation from Shaanxi Institute of Mechanical Engineering of China, Xi'an in 1992, M.S. degree in computer application technology from Xi'an University of Technology of China, Xi'an in 1999, and Ph.D. degree in computer software and theory from Northwest University of China, Xi'an in 2002. He was in University of Tsukuba of Japan between March to September 2004 as a visiting scholar. He is currently a professor with School of Computer Science and Engineering, Xi'an University of Technology, China. His research interests include Internet of Things technology, network computing.

Jinfa Shi male, doctor, professor, master tutor. He was born in Qidong city, Jiangsu province, P. R. China in January, 1963. He received his B.Sc. and M.Sc. both from Chongqing Jianzhu University in 1987 and 1990, and the doctor degree in State Key Laboratory of Mechanical Drives of Chongqing University in 1994. Now he is a professor and the master tutor of School of Management Science and Engineering, Zhengzhou Institute of Aeronautical Industry Management. His major research interests include cloud computing and information management.

Jian Li male, doctor, professor. He was born July 16, 1962. He received bachelor degree in Department of Mathematics of Henan University in 1984, the master degree in School of Computer and Information Engineering of Henan University in 2003. He is currently a professor with School of Computer, Henan Institute of Engineering, China. His research interests include computer network, network security.