International Journal of Grid and Distributed Computing
Vol. 9, No. 4 (2016), pp.161-176
http://dx.doi.org/10.14257/ijgdc.2016.9.4.15

An Improved Multi-Objective Optimization Algorithm Based on NPGA for Cloud Task Scheduling

Peng Yue [1], Xue Shengjun [2], Li Mengying[3]

[1,2,3]*School of Computer&Software, Nanjing University of Information Science&Technology, Nanjing 210044, China*
School of Computer&Software, Nanjing University of Information Science&Technology, Jiangsu Engineering Center of Network Monitoring, Nanjing 210044, China
[1] *18751972517@163.com*, [2] *sjxue@163.com*, [3] *lmy3612@163.com*

Abstract

As a commercial distributed computing mode, cloud computing needs to meet the quality of service (QoS) requirement of users, which is its top priority. However, cloud computing service providers also need to consider how to reduce the overhead of data center, and keep load balancing is one of the key points to maximize the use of the resource in the data center. In this paper, we propose an improved multi-objective niched Pareto genetic algorithm (NPGA) to take load balancing into consideration without affecting performance of time consumption and financial cost of handling the user's cloud computing tasks by presenting the load balancing shift mutation operator. The simulation results and analysis show that the proposed algorithm performs better than NPGA in maintaining the diversity and the distribution of the Pareto-optimal solutions in the cloud tasks scheduling under the same population size and evolution generation.

Keywords*: Cloud Task Scheduling; improved NPGA; Load Balancing shift mutation operator; Three-objectives optimization*

1. Introduction

With the progress of the digitization process in the society, huge amounts of data are continuously generated every day. How to store and analyze massive amounts of data have efficiently become hot issues. In this time, the development of big data theory and cloud computing technology offers powerful solutions for the issues [1]. And it is becoming an increasingly common trend to handle massive data by using cloud computing.

As a commercial distributed computing mode, cloud computing is different from grid computing, it pays more attention to the quality of service (QoS) requirement of users [2]. The demands of the cloud users are diverse: some users may want their tasks to be handled quickly though they need to pay more money while others don't want to pay so much money for dealing their tasks because they do not care about the time consumption. Therefore the scheduling models which just take time consumption as constraint and financial cost as optimization objective or vice versa can't satisfy cloud computing well. Besides, a good cloud computing algorithm also needs to take load balancing of the data center into consideration so that the resource in the data center can be fully utilized.

Therefore, we propose an improved niched Pareto genetic algorithm (NPGA) which takes the time consumption and financial cost of computing the users' tasks and load balancing of virtual machines in the data center as the objectives in the multi-objective optimization of cloud tasks scheduling. The similar task order crossover (STOX) operator [3] is applied in the algorithm so that the algorithm can evolve more efficient while load

ISSN: 2005-4262 IJGDC

balancing shift mutation operator is applied in the progress of evolution to avoid the premature convergence and ensure the load balancing at the same time.

2. Scheduling Model

2.1. Mathematical Model

Cloud computing is based on distributed computing and it can handle multiple tasks at the same time [4]. The relationship between these tasks can be independent or dependent, which forms a hybrid cloud task model and makes the establishing process of the task model more complicated. In order to describe the model clearly, we use directed acyclic graph (DAG) to form a cloud task model which contains both independent tasks and tasks with dependency relationship at the same time in this paper as shown in Figure. 1. The hybrid cloud task model differs from the workflow task model [5] [6] or independent task model [7][8], the constraints during the process of the task scheduling are more complicated.

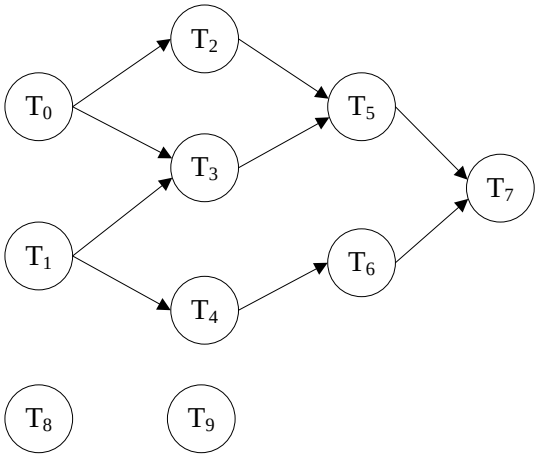

Figure 1. Hybrid Cloud Task Model

As for the cloud tasks, they need to be handled by the virtual machines in the data center according to the specific task scheduling strategy, so the mapping relations between the cloud tasks and the virtual machines are the key points which will affect the result of the cloud computing task scheduling. We use a collection of $S=\{<T>,<V>,<M>\}$ to define the elements in the progress of cloud computing task scheduling. In this collection, $<T>$ represents the set of cloud tasks, $<V>$ represents the set of virtual machines and $<M>$ represents the set of mapping relations between the cloud tasks in $<T>$ and virtual machines in $<V>$.

In cloud computing, the users' main concerns are the time consumption and the financial cost of handling the tasks they submit to the cloud data center. However, the cloud computing service provider also pay attention to the load balancing of the cloud computing tasks on the virtual machines in the cloud data center. So we take the time consumption, financial cost of handling the tasks which the users submit to the cloud data center and the load balancing of the cloud tasks on the virtual machines in the cloud data center as the multiple optimization objectives.

In this paper, we focus on how to allocate the cloud tasks to the virtual machines to make sure the Pareto-optimal front describing the performance of the multiple optimization objectives can approach to the real optimal front much closer. Also, we need to maintain the diversity and the distribution of the Pareto-optimal solutions in the cloud tasks scheduling. In order to achieve the goal mentioned above, a set of assumptions about the details in the collection of $S=\{<T>,<V>,<M>\}$ need to be proposed before

the objective function is established.

(1) Every cloud task node in $<T>$ can be allocated to the virtual machine in $<V>$ only when all the tasks on which it depends have been processed completely.
(2) The virtual machine in $<V>$ can only process one cloud task in $<T>$ at a time and the virtual machine can not be occupied before the cloud task is processed completely. Moreover, if there are more than one task waiting for be processed, then the task with less amount of computation has higher priority to occupy the virtual machine.
(3) During the progress of handling the cloud tasks on the virtual machines in the cloud data center, the mapping relation in $<M>$ concerns about the objectives of the time consumption, the financial cost and the load balancing situation mentioned above.

2.2. Objective Function

With these assumptions, we can establish the objective function now. If the number of the cloud tasks is p and the number of the virtual machines is q, then M_{ij} in $<M>$ represents the cloud task T_i in $<T>$ is allocated to the virtual machine V_j in $<V>$, and $0 \le i \le p-1, 0 \le j \le q-1$. According to the assumption (1) in the mathematical model part, we define $Pre(T_i)$ as the collection of direct predecessors of T_i, T_i only can be allocated to specific virtual machine when $Pre(T_i)$ is empty. As to the constraint in assumption (2), we define $Col(T_i)$ as the collection of tasks which are allocated to the same virtual machine with task T_i and the task in $Col(T_i)$ will has higher priority if it has less amount of computation. So T_i can occupy the virtual machine only when all the tasks in $Col(T_i)$ are processed completely, that is, $Col(T_i)$ is empty. What's more, according to the assumption (3), we define t_{total} as the time consumption of handling all the tasks submitted to the cloud data center, c_{total} as the financial cost of using all the resources in the cloud data center, LB as the load balancing degree of the cloud tasks scheduling on the virtual machines.

As mentioned above, the start time of processing T_i depends on the situation of $Pre(T_i)$ and $Col(T_i)$, if there are *m* task nodes in $Pre(T_i)$ and *n* task nodes in $Col(T_i)$, then the start time of processing T_i can be defined as $t_s(T_i)$ in formula (1).

$$t_s(T_i) = \begin{cases} 0, & Pre(T_i) = \varphi, Col(T_i) = \varphi \\ \max\limits_{0 \le k \le m-1} t_e(T_k), & Pre(T_i) \ne \varphi, Col(T_i) = \varphi \\ \sum\limits_{j=0}^{n-1} t_e(T_j), & Pre(T_i) = \varphi, Col(T_i) \ne \varphi \\ \max\limits_{0 \le k \le m-1} t_e(T_k) + \sum\limits_{j=0}^{n-1} t_e(T_j), & Pre(T_i) \ne \varphi, Col(T_i) \ne \varphi \end{cases} \quad (1)$$

As shown in formula (1), $t_e(T_k)$ represents the finish time of the last task in $Pre(T_i)$ and $0 \le k \le m-1$, as for $t_e(T_j)$, it represents the finish time of the last task in $Col(T_i)$ and $0 \le j \le n-1$.

With knowing the start time of processing task T_i, we can define the finish time of processing task T_i with formula (2).

$$t_e(T_i) = \frac{amount(T_i)}{ability(V_j)} + t_s(T_i) \quad (2)$$

In formula (2), $amount(T_i)$ represents the amount of computation of task T_i, $ability(V_j)$ represents the computation ability of virtual machine V_j which the task T_i is allocated to.

We define T_{last} as the last task which is processed completely on the virtual machines while T_{first} is the first one. As mentioned above, the number of the cloud task is p, so $last - first + 1 = p$ and the total time consumption of handling the cloud tasks t_{total} can be described in formula (3) as follows.

$$t_{total} = t_e(T_{last}) - t_s(T_{first}) \tag{3}$$

As for the total financial cost c_{total}, it can be described as the formula (4). In formula (4), c_{ij} means the unit price of using the virtual machine V_j which task T_i is allocated to, and c_{ij} will be much higher if the computation ability of the virtual machine is much stronger.

$$c_{total} = \sum_{i=1}^{p} c_{ij} * (t_e(T_i) - t_s(T_i)) \tag{4}$$

The load balancing degree mentioned in this paper is about the situation of mapping relations between the cloud tasks and the virtual machines. We define VL_j as the total execution time that V_j needs when it handles all the tasks allocated to it one by one without any other constraints. In formula (5), u represents the number of tasks which are allocated to the virtual machine V_j.

$$VL_j = \sum_{i=0}^{u-1} \left(\frac{amount(T_i)}{ability(V_j)}\right) \tag{5}$$

So the average execution time of all the tasks on the virtual machines can be defined as $\overline{VL}$ in formula (6) and w represents the number of virtual machines which are occupied by the cloud tasks.

$$\overline{VL} = \frac{\sum_{j=0}^{w-1} VL_j}{p} \tag{6}$$

Then, the load balancing degree LB can be described as follows in formula (7).

$$LB = \frac{\sqrt{\sum_{j=0}^{w-1} (VL_j - \overline{VL})^2}}{w} \tag{7}$$

In this paper, we choose the total time consumption t_{total}, the total financial cost c_{total} and the load balancing degree LB as the multiple optimization objectives. For the users who pay for the access of the resources in the cloud data center, they do want t_{total} to be much shorter while c_{total} is much cheaper, and for the cloud computation service providers, they also care about whether their resources are widely used, they hope the load balancing degree LB can be much lower which means the cloud tasks are widely allocated on the virtual machines. So the multi-objective function can be established as follows in formula (8).

$$\begin{cases} \min(t_{total}) = \min(t_e(T_{last}) - t_s(T_{first})) \\ \min(c_{total}) = \min(\sum_{i=1}^{p} c_i * (t_e(T_i) - t_s(T_i))) \\ \min(LB) = \min(\frac{\sqrt{\sum_{j=0}^{w-1} (VL_j - \overline{VL})^2}}{w}) \end{cases} \quad (8)$$

2.3. Encoding Design of the Mapping Relations

As for the model mentioned above, it has p cloud tasks and q virtual machines, so we can use a table to describe the mapping relations between the cloud tasks and the virtual machines in $<M>$ as shown in table 1. The table represents a matrix A with q rows and p columns, we take a_{ij} as the element at row i and column j in A and $0 \le i \le q-1$, $0 \le j \le p-1$. All the elements in A can only be assigned to 1 or 0. For example, if the cloud task T_j is allocated to the virtual machine V_i, then a_{ij} =1, otherwise a_{ij} =0.

Table 1. Encoding

V_i \ T_j	T_0	T_1	T_2	…	T_{p-1}
V_0	0	1	0	…	1
V_1	1	0	0	…	0
V_2	0	0	0	…	0
⋮	⋮	⋮	⋮		⋮
V_{q-1}	0	0	1	…	0

As for the cloud task T_j, it can only be allocated to one virtual machine to be processed, so at most there is one element can be assigned to 1 in every column in table 1. However, one virtual machine can receive more than one cloud tasks, so more than one element can be assigned to 1 in every row in table 1.

For every single independent task in $<T>$, its execution time will be much shorter if it is allocated to the more powerful virtual machine. However, the powerful virtual machine can only process one cloud task at the same time, they have to wait for some time to occupy the computation resources while the other virtual machines are free if too many cloud tasks are allocated to the same virtual machine, so the total time consumption t_{total} may not be the shortest if the cloud tasks are scheduled with this solution. Meanwhile, the unit cost of using the virtual machine will be more expensive if the computation ability of the virtual machine is more powerful, obviously, it's not wise to allocate too many tasks to the expensive virtual machines. What's more, both the number of the virtual machines which the cloud tasks are allocated to and the mapping relations between the cloud tasks and these virtual machines will affect the load balancing degree LB. So it's complicated to find a scheduling solution encoding which can meet the demand of the multi-objective function in formula (8) to make the time consumption

t_{total}, the financial cost c_{total} and the load balancing degree LB achieve the Pareto-optimal values in the Pareto-optimal front.

3. An Improved NPGA Algorithm for Cloud Task Scheduling

3.1. Basic Algorithm

The encoding process mentioned above can be used to form the chromosome in NPGA, and the population with this kind of chromosomes will go through several rounds of evolution to get the final population in which the chromosomes are the solutions in the Pareto-optimal front. The main steps of the basic NPGA are as follows [9].

Step 1: Initialize the population $P(gen)$ and select the chromosomes randomly from $P(gen)$ to form a comparison set (CS) of which the size is t_{dom}.

Step 2: Two chromosomes are selected randomly from the population $P(gen)$, we take the two chromosomes as p_i and p_j. If p_i is not dominated by any chromosome in CS and p_j is dominated by at least one chromosome in CS, then p_i is much fitter than p_j and p_i will be selected into next generation directly. On the contrary, p_j will be selected into the next generation. If p_i and p_j are both not dominated by any chromosome in CS or p_i and p_j are both dominated by at least one chromosome in CS, then which one of p_i and p_j should be selected into the next generation can not be decided by only using CS, p_i and p_j will go through the calculation of niche count.

Step 3: Take the niche count of p_i as nc_i and the niche count of p_j as nc_j. After getting the value of nc_i and nc_j, we can know which one of p_i and p_j is much fitter by comparing nc_i and nc_j. If $nc_i \le nc_j$, then p_i is much fitter than p_j, we select p_i into the next generation. Otherwise, p_j is much fitter than p_i and p_j will be selected into the next generation.

Step 4: Repeat the steps from Step 2 to Step 3 until all the chromosomes in current population have taken part in the comparison.

Since the processes in Step 3 are complicated, we will do a brief introduction of the calculation as follows.

Taking p_i for example, it will be put into the next generation $P(gen+1)$ which is not filled up at the moment when p_i is selected. We will calculate $d(i,k)$ which represents the Euclidean distance [10] between every single chromosome $p_k \in P(gen+1)$ and p_i. Within the range of niche radius σ_{share}, the niche count of p_i is defined as nc_i in formula (9).

$$nc_i = \sum_{k \in P(gen+1)} sh[d(i,k)] \tag{9}$$

In formula (9), $sh[d(i,k)]$ is the sharing function, it is defined as follows in formula (10).

$$sh[d(i,k)] = \begin{cases} 0 & d(i,k) > \sigma_{share} \\ 1 - d / \sigma_{share} & d(i,k) \le \sigma_{share} \end{cases} \tag{10}$$

3.2 Improved Strategies

NPGA is efficient when dealing with some issues, and it is necessary to reduce the time of occupying the computing resource of namenode when it is applied to the cloud task scheduling problem since the computing resource of namenode is precious[11]. Also, the diversity of individuals is another aspect the scheduling algorithm should pay attention to.

(1) Similar Task Crossover Operator

STOX can help the algorithm converge faster, we will use a scheduling model with 10 cloud tasks and 3 virtual machines to describe the main steps it contains [12].

Step 1: Copy the common subsequences from parents to offspring.

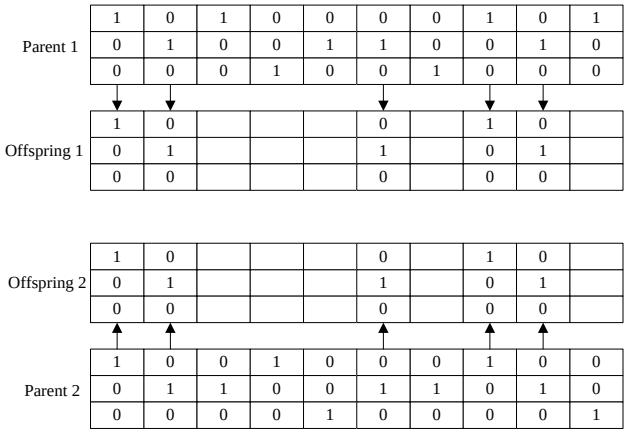

Figure 2. Operation in Step 1

Step 2: Inherit the subsequences before the cut point from their respective parents.

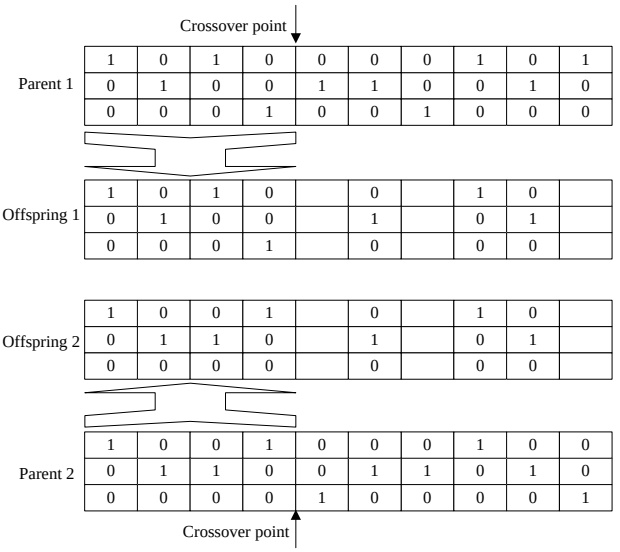

Figure 3. Operation in Step 2

Step 3: Copy the missing subsequences form the parents of their respective companions.

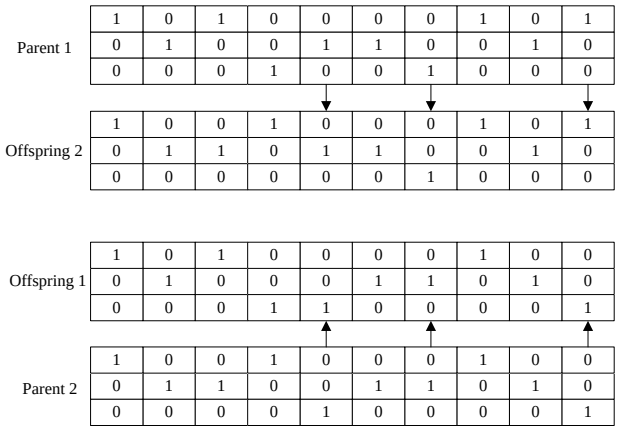

Figure 4. Operation in Step 3

By doing the steps mentioned above, the excellent subsequences of the gene in the chromosome during the process of revolution can be maintained to improve the rate of convergence.

(2) Load Balancing Shift Mutation Operator

The STOX operator can make the scheduling algorithm achieve the Pareto-optimal front faster. However, it may also cause the algorithm to converge to the local optimal solutions, so the load balancing shift mutation (LBSM) operator is applied to avoid the phenomenon. What's more, the LBSM operator can take care of the load balancing degree of the distribution of the cloud tasks on the virtual machines.

The LBSM operator mainly includes the following steps.

Step 1: The start point and the end point are selected randomly with a fixed step length.

Step 2: Choose a specific column between the start point and the end point in sequence, check the distribution of the cloud tasks on the virtual machines according to the subsequences which are definite.

Step 3: Calculate the probabilities of allocating the cloud task corresponding to the column selected in step 2 to every other virtual machine, and then relocate the cloud task to the new virtual machine by using the roulette wheel selection[13].

Step 4: Check whether all the columns between the start point and the end point have been through the process of relocation operation, if yes, the process of LBSM operator is completed, otherwise go to Step 2.

We still use a scheduling model with 10 cloud tasks and 3 virtual machines to describe the main steps LBSM contains. As shown in Figure. 5, the step length is 2, so there are two columns between the start point and the end point, which means two cloud tasks need to be relocated by using the LBSM operator.

					Start Point		End Point		
1	0	1	0	0	0	1	1	0	1
0	1	0	0	1	1	0	0	1	0
0	0	0	1	0	0	0	0	0	0

Figure 5. The Chromosome before the LBSM

We choose the first column between the start point and the end point from left to right and check the subsequences which are definite, so we can calculate the probabilities now as shown in table 2.

Table 2. The Calculation of the Cumulative Probability of being Chosen

Virtual Machine	V1	V2	V3
The Number of Allocated Tasks	4	3	1
The Probability of Being Chosen	3/19	4/19	12/19
The Cumulative Probability of Being Chosen	3/19	7/19	1

After getting the cumulative probabilities in table 2, we just need a random number to determine which virtual machine the cloud task will be relocated to. Assume the random number in the roulette wheel selection to be 0.6, so the chosen virtual machine in this round will be V3. Update the list of definite subsequences and choose the second column between the start point and the end point from left to right. As for every other virtual machine in the definite subsequences, the cumulative probabilities of being chosen are shown in Table 3.

Table 3. The Calculation of the Cumulative Probability of being Chosen

Virtual Machine	V1	V2	V3
The Number of Allocated Tasks	4	3	2
The Probability of Being Chosen	3/13	4/13	6/13
The Cumulative Probability of Being Chosen	3/13	7/13	1

Assume the random number in the roulette wheel selection to be 0.4, so the chosen virtual machine in this round will be V2.

The LBSM operator is completed now and the encoding after the operator is shown as Figure. 6.

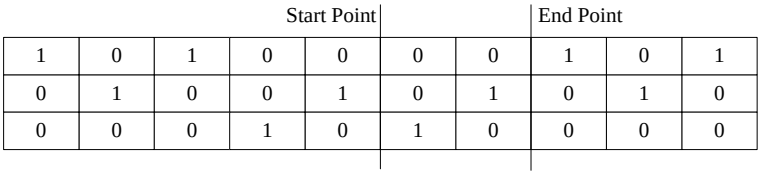

Figure 6. The Chromosome after the LBSM

By doing the LBSM, the load balancing of the cloud tasks on the virtual machines is improved. Meanwhile, the LBSM operator helps the algorithm avoid the premature convergence.

(3)Self-adapting Tournament Selection

In addition, most of the chromosomes in the population are in the Pareto-optimal front at the later stage of the revolution and there are no dominating relations between these chromosomes, so it's useless to compare the chromosomes by using CS at this moment. Traditionally, the size of CS, that is, t_{dom}, ranges from 2 to M which represents the size of the population[14]. So it is reasonable that t_{dom} should be designed to decrease linearly with the increase of the generation of the evolution.

On the other hand, the niche radius in the traditional NPGA takes the mean value of the Euclidean distance between the adjacent individuals in the Pareto-optimal front. However, if the distribution of the individuals in the early stage is quite different from what in the late stage when filling the next generation population, obviously, there will be serious deviation in the selection of outstanding individuals by using the niche radius in the early stage. So a variant of the $sigmoid$ function [15] is applied to make the niche radius

σ_{share} vary with the rate of progress of filling the next generation so that the deviation in the selection in the early stage can be effectively reduced and the diversity of individuals in the Pareto-optimal front can be improved significantly. The value of σ_{share} is set according to the formula (11) as follows.

$$\sigma_{share}(fill) = d_{\min} + (d_{avg} - d_{\min})(1 + \exp(-a\frac{fill}{Pop})) \quad (11)$$

In formula (11), $fill$ represents the number of the individuals which have been selected into the next generation population, $d_{\min}$ represents the minimum value of the Euclidean distances between the adjacent individuals which have been selected into the next generation population while d_{avg} represents the mean value of it. What's more, a is the adjustable parameter which is responsible for adjusting the altering trend of σ_{share} and Pop represents the size of the population which is set in the initialization.

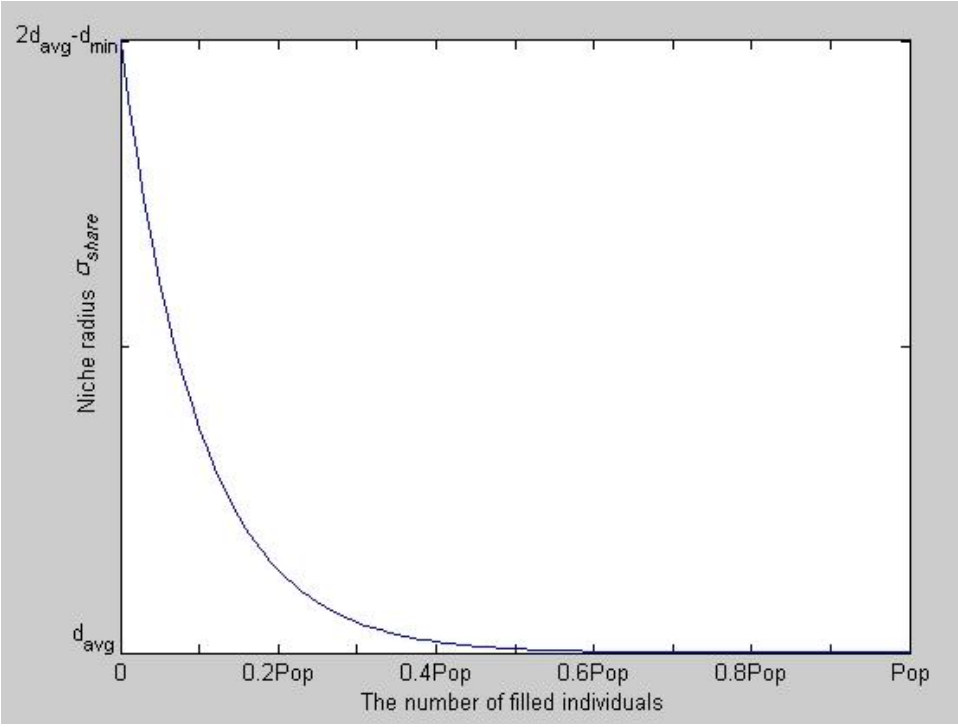

Figure 7. The Adaptive Niche Radius Curve

According to formula (11), the niche radius σ_{share} trends to $2d_{avg} - d_{\min}$ in the early stage of the filling the next generation population, the value is slightly larger so that the loss of the outstanding individuals can be mitigated. Gradually, as the next generation of the population is filled to a certain extend, the average value of the Euclidean distances between the adjacent individuals can well reflect the distribution of the individuals in the population of this generation, so σ_{share} trends to d_{avg} at this stage. Experiments show that when a values 10, σ_{share} will flatten out gradually with $fill$ getting near to $0.4\,Pop$, which is responsible as shown in Figure. 7.

4. Experiments and Analysis

CloudSim is an open source cloud computing simulator developed and maintained by the team led by Professor Rajkumar Buyya in the University of Melbourne [16]. In this paper, we use this software to implement the simulation of the algorithms in cloud computing. After the simulation, we compare the distribution of the individuals in the Pareto-optimal fronts generated by the traditional NPGA and the improved NPGA. Also, the influence of population size is taken into consideration. Finally, the comparison of the Pareto-optimal fronts is illustrated so that the advantage of the improved NPGA can be seen clearly.

4.1. The Performance Comparison between the Basic Algorithm and the Improved Algorithm

As for the cloud task scheduling algorithms, the larger the number of the non-repetitive individuals in the Pareto-optimal front is, the stronger the ability of finding the Pareto-optimal solutions is. On the other hand, the distribution of the individuals in the Pareto-optimal front is measured by the mean value and the stand deviation of the distances between the adjacent individuals, the smaller these values are, the better the distribution is.

We take a model with 10 cloud tasks and 8 virtual machines for example. The population ranges from 100 to 500 with the step size 100 as well as the evolution generation and the improved NPGA trends to stability when the population is set to 300, so we take the comparison of the algorithms in such a situation as shown in Figure. 8, Figure. 9 and Figure.10.

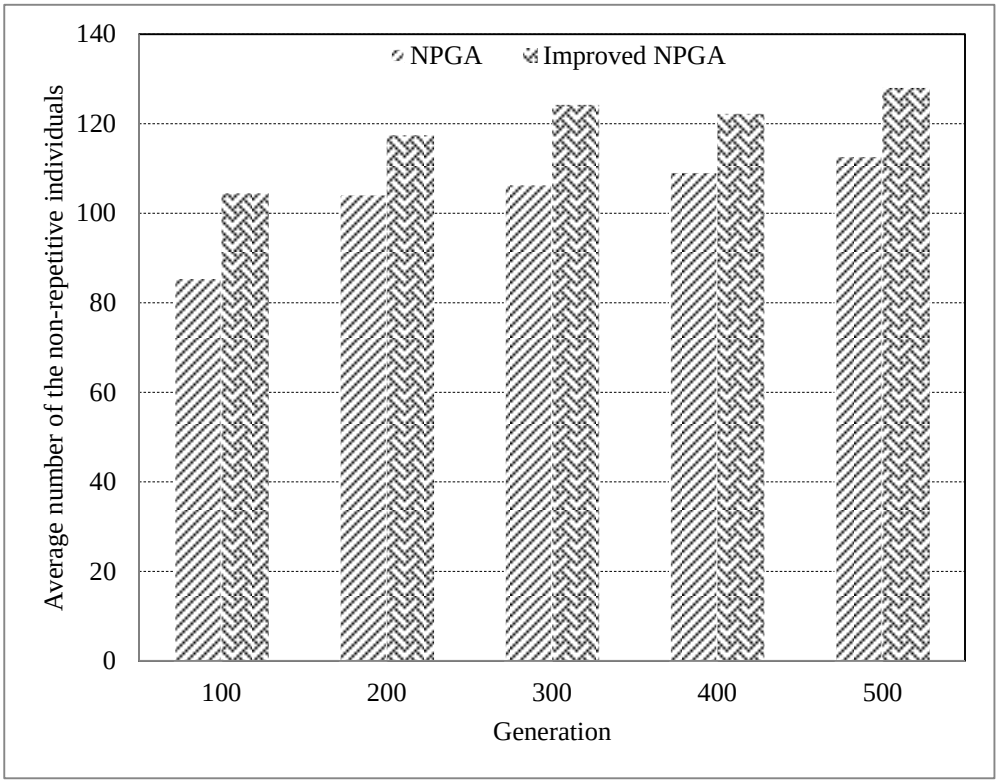

Figure 8. The Number of the Non-repetitive Individuals in the Pareto-Optimal Front

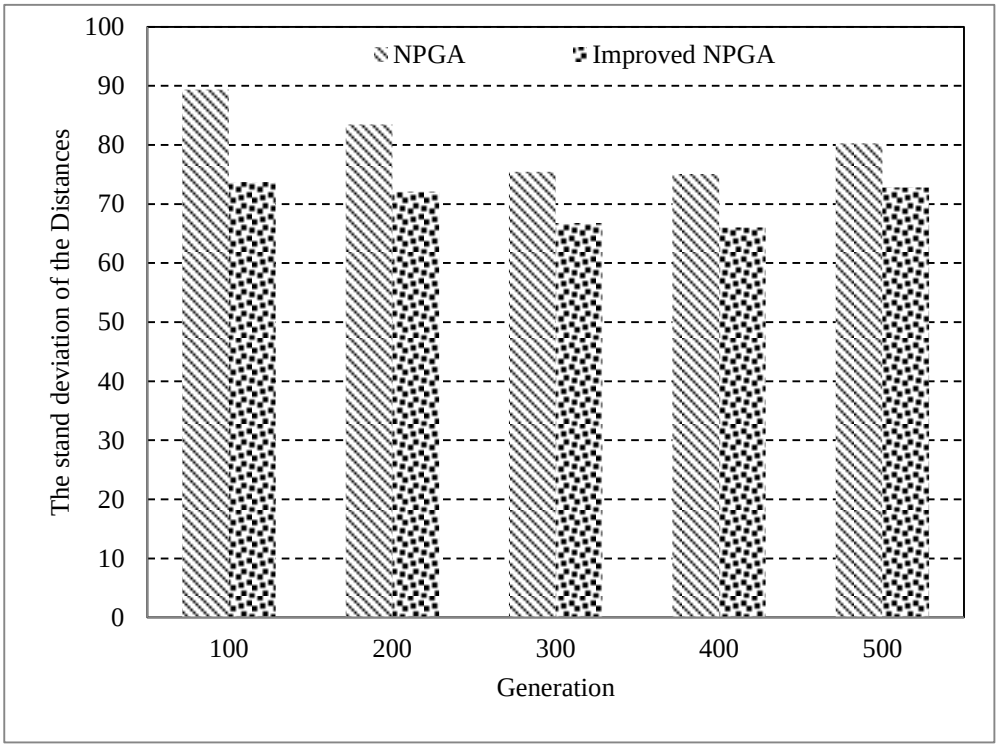

Figure 9. The Stand Deviation of the Distances between the Adjacent Individuals

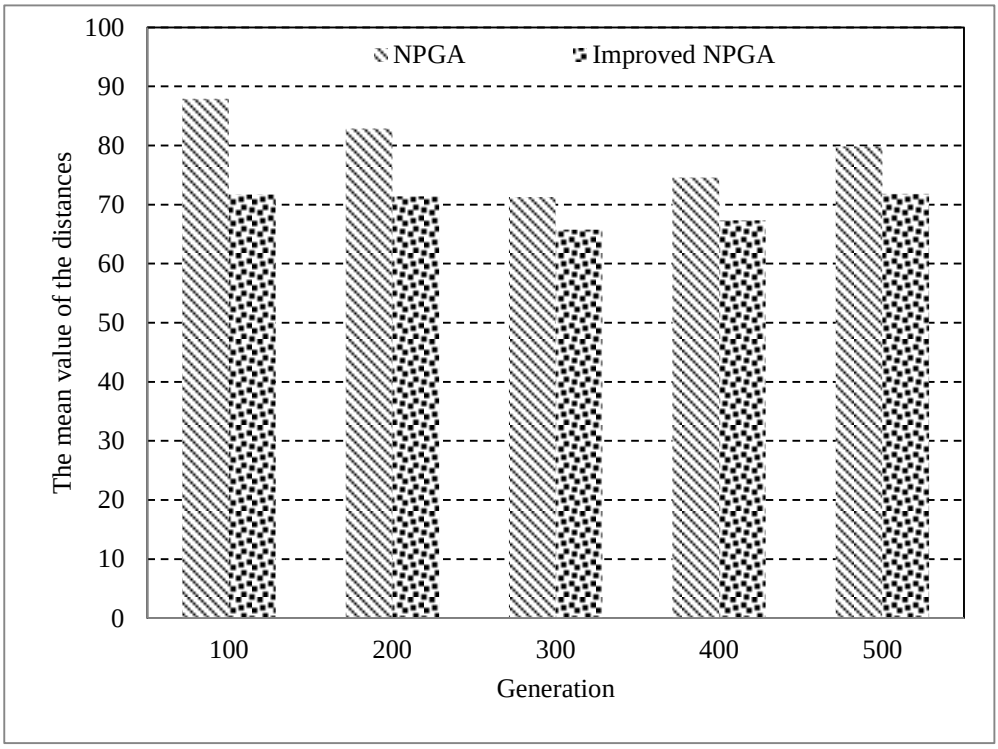

Figure 10. The Mean Value of the Distances between the Adjacent Individuals

We can see from Figure. 8 that the improved NPGA always finds more

Pareto-optimal individuals than the traditional NPGA when they are under the same situation, which means the improved NPGA has stronger ability to find the outstanding solutions for the cloud task scheduling problem. On the other hand, Figure. 9 and Figure. 10 show that the stand deviation and the mean value of the distances between the adjacent individuals in the Pareto-optimal front which is generated by the improved NPGA are smaller, which means the distribution of the individuals in the Pareto-optimal front is more even and more continuous.

4.2. The Influence of Population Size on the Performance of the Algorithm

NPGA is essentially an evolutionary algorithm, so both the population size and the evolution generation will affect the performance of the final set of optimal solutions obtained by the Improved NPGA. Since we have discussed the influence of the evolution generation on the performance in the previous section, we will see how the population size will affect the algorithm when it goes through enough evolution generations. We still take the cloud task model mentioned above as example. By running the improved algorithm 50 times, we take the mean value of the result to see the influence of population size on the performance of the algorithm as shown in table 4.

Table 4. Influence of Population Size on the Algorithm

Population size	The average number of solutions	Mean value	Stand deviation
100	55.4	120.0315	91.7037
200	105.8	78.5643	74.0702
300	124.2	65.7768	66.7963
400	128.0	69.8393	72.7061
500	126.6	70.8442	74.2710

We can see from the table that the appropriate population size is important to the improved NPGA. If the population size is too small, the algorithm will not obtain a Pareto-optimal front which can contain almost all the excellent individuals. However, if the population size is too large, the algorithm can not get more excellent individuals while the computing resource of the host machine will be occupied by the algorithm for a long time, which should be avoided in cloud computing because the resource of the namenode is so precious that it even becomes the bottleneck of cloud computing [11].

4.3. Pareto-Optimal Front Comparison

We select the time consumption, the financial cost and the load balancing degree as the multiple optimal objectives in the cloud task scheduling in this paper, and every individual in the Pareto-optimal front represents an available solution which meets the demand of achieving the Pareto-optimal values in all the aspects of the optimization model of the cloud task scheduling. Since the optimal individuals range widely, we choose part of the Pareto-optimal front to describe the differences of the performance of the traditional NPGA and the improved NPGA so that we can see the differences much more clearly. As mentioned above, all the optimization objectives pursue small values as the excellent result so we can see from Figure.11 that the performance of the improved NPGA is much better than the traditional NPGA when they are under the same population size and the same evaluation generation.

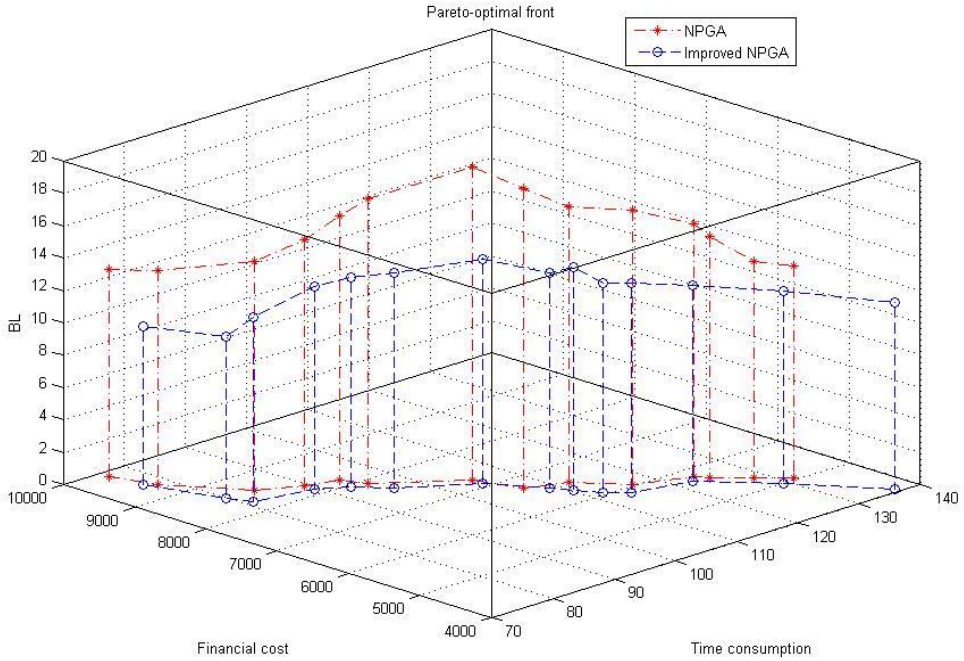

Figure 11. The Comparison of Part of the Pareto-Optimal Fronts

5. Conclusion and Future Works

In this paper, we propose a cloud task model which is made up of independent tasks and workflow tasks. Based on the hybrid cloud task model, we select the time consumption, the financial cost and the load balancing of processing the cloud tasks on the virtual machines as the optimization objectives. In addition, the NPGA is applied to the cloud task scheduling problem and we use the STOX operator, LBSM operator and the self-adapting tournament selection operator to improve the algorithm so that the evolution of the improved algorithm is more efficient and the diversity of the individuals in the Pareto-optimal front of the improved algorithm is much better than the traditional NPGA.

However, we assume the cloud tasks are computation-intensive in this paper, so we ignore the influence of the bandwidth and the I/O delay on the time consumption and the financial cost to simplify the cloud task scheduling model. In the future, we should take more details of the cloud task scheduling into consideration to make sure the result of the simulation can approach the real situation closer.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (Grant Nos.41275116), Jiangsu Economic and Information Technology Commission project of China(Grant Nos.{2011}1178) and Open Fund Project of Jiangsu Engineering Center of Network Monitoring (Grant Nos.KJR1309).

References

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh and McKinsey Global Institute, “Big data: The next frontier for innovation, competition, and productivity”, **(2011)**.
[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, M .Lindner, “A break in the clouds: towards a cloud definition”, ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, **(2008)**, pp. 50-55.

[3] S. J. Xue, F. Liu, X. L. Xu, "An Improved Algorithm Based on NSGA-II for Cloud PDTs Scheduling", Journal of Software , vol. 9, no. 2, **(2014)**.
[4] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, A. Vakali, "Cloud computing: Distributed internet computing for IT and scientific research", Internet Computing, IEEE, vol. 13, no. 5, **(2009)**, pp. 10-13.
[5] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, "On the use of cloud computing for scientific workflows", the fourth International Conference on IEEE, **(2008)**, pp. 640-645.
[6] S. Pandey, L. Wu, S. M. Guru, R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", the 24th IEEE International Conference in Advanced Information Networking and Applications, **(2010)**, pp. 400-407.
[7] W. Shi, B. Hong, "Resource allocation with a budget constraint for computing independent tasks in the cloud", the Second International Conference on Cloud Computing Technology and Science, **(2010)**, pp. 327-334.
[8] H. Zhu, Y. Wang, L. Fan, X. Wang, "Grid Independent Task Scheduling Multi-Objective Optimization Model and Genetic Algorithm", Journal of Computers, vol. 5, no.12, **(2010)**.
[9] M. Erickson, A. Mayer, J. Horn, "Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA)", Advances in Water Resources, vol. 25, no. 1, **(2002)**, pp. 51-65.
[10] P. Legendre, E. D. Gallagher, "Ecologically meaningful transformations for ordination of species data", Oecologia, vol. 129, no. 2, **(2001)**, pp. 271-280.
[11] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The hadoop distributed file system", the 26th Symposium on IEEE of Mass Storage Systems and Technologies, **(2010)**, pp. 1-10.
[12] Ruiz, Rubén, Concepción Maroto, and Javier Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem", Omega vol. 34, no. 5, **(2006)**, pp. 461-476.
[13] D. E. Goldberg, K. Deb, "A comparative analysis of selection schemes used in genetic algorithms", Urbana, **(1991)**.
[14] M. A. Abido, "A niched Pareto genetic algorithm for multi-objective environmental/economic dispatch", International journal of electrical power & energy systems, vol. 25, no. 2, **(2003)**, pp. 97-105.
[15] X. Yin, J. A. N. Goudriaan, Lantinga E A, J. A. N. Vos, H. J. Spiertz, "A flexible sigmoid function of determinate growth", Annals of Botany, vol. 91, no. 3, **(2003)**, pp. 361-371.
[16] R. N. Calheiros, R. Ranjan, A. Beloglazov, De Rose, C. A., Buyya, R, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, vol. 41, no. 1, **(2011)**, pp. 23-50.

Authors

Peng Yue, was born in Jiangsu, China. She received the B.S. degree in 2015 in network engineering from Nanjing University of Information Science & Technology, Nanjing, China. She is now studying as a M.S. at Nanjing University of Information Science & Technology, Nanjing, China.Her research interests in cloud scheduling, resource allocation and genetic algorithm.

Xue Shengjun, was born in Qingdao, China. He received the B.S. degree in 1983 in computer science & technology from Zhejiang University, Hangzhou, China. He received the M.S. degree in 1998 and the Ph.D. degree in 2002 from Wuhan University of Technology, Wuhan, China. He once worked as a postdoctoral researcher at Purdue University.Now he is the professor of computer science in Nanjing University of Information Science & Technology, Nanjing, China. His research interests in network of computer, cloud computing, intelligent transport and applied meteorology.Prof. Xue is a member of IEEE, and he is also the advanced member of Chinese Computer Federation (CCF).

Li Mengying, was born in Nanjing, China. She received the B.S. degree in 2013 in network engineering from Nanjing University of Information Science & Technology, Nanjing, China. She is now studying as a M.S. at Nanjing University of Information Science & Technology, Nanjing, China.Her research interests in cloud scheduling, genetic algorithm and multi-objective optimization.