An Improved Algorithm Based on NSGA-II for Cloud PDTs Scheduling

Shengjun Xue
Nanjing University of Information Science & Technology, School of Computer and Software, Nanjing, China
Nanjing University of Information Science & Technology, Jiangsu Engineering Center of Network Monitoring, Nanjing, China
Email: nuist_lf@163.com

Fei Liu, Xiaolong Xu
Nanjing University of Information Science & Technology, School of Computer and Software, Nanjing, China
Email: {jkleo07@126.com, xlxu1988@gmail.com}

***Abstract*—Partly dependent tasks (PDTs) scheduling with multi-objective optimization in cloud computing is an NP-hard problem. Taking the quality of service (QoS) requirements of users that use cloud computing into account, we set the cost and time requirements of handling the PDTs as the multiple objectives and present an improved algorithm based on the non-dominated sorting genetic algorithm-II (NSGA-II) to find the Pareto optimal set of the PDTs scheduling. In this paper, the similar task order crossover (STOX) operator is applied to make the evolution more efficient while the shift mutation operator is applied in the process of evolution to avoid the premature convergence. In addition, we present a new method named self-adapting crowding distance (SCD) operator, which can improve the diversity of individuals in the Pareto-optimal front. The simulation results and analysis show that the proposed algorithm performs better than NSGA-II in maintaining the diversity and the distribution of the Pareto-optimal solutions in the cloud PDTs scheduling.**

***Index Terms*—PDTs, multi-objective optimization, cloud computing, STOX, SCD, improved NSGA-II**

I. INTRODUCTION

As a new commercial network computing mode, cloud computing has attracted many users' attention because of its good performance in processing large dataset. Besides focusing on computational efficiency and throughput, the cloud computing service provider pays more attention to user experience since the users pay for the storage and computing resources when they apply for the cloud computing service [1][2].

The users don't care about how the tasks they submitted were processed, it makes the relations between the tasks more complicated than those in the traditional distributed system [3][4]. Previous studies are mostly based on independent tasks scheduling [5][6] or workflow tasks scheduling [7][8], but these kinds of models can't simulate the complicated relations between the tasks in the cloud computing environment.

Moreover, for most users, the cost and the time consumed when their tasks are processed in the data center of the cloud computing service provider are the two most important factors of their interest. There are many researches about getting the extreme value in one aspect by setting a limit in another. However, these kinds of researches tend to neglect the fact that the users may not have clear impressions on their budgets about the time or financial cost, it may be impractical to request the users to submit definite limitations, and it may be much better if the service provider can offer the users some solutions so they can choose one which is appropriate for them in both aspects of the requirements of time and cost. Obviously, the results of researches mentioned earlier can't meet the requirement.

In this paper, we introduce a new scheduling model, that is the partly dependent tasks (PDTs) scheduling model [9], and the PDTs are composed of independent tasks and workflow tasks. In order to get a solution fulfilling the demands of both the time consumption and the financial cost budgets of the PDTs scheduling, the non-dominated sorting genetic algorithm-II (NSGA-II) [10][11] is applied in the process of task scheduling. However, this density-based algorithm doesn't converge fast in finding the Pareto-optimal front, moreover, the individuals in the Pareto-optimal front are not well distributed. As a consequence the similar task order crossover (STOX) [12], shift mutation [13] and self-adapting crowding distance (SCD) operators are adopted in the improved NSGA-II to settle these problems.

The rest of this paper is organized as follows. In Section II, we describe the scheduling model with the fitness function which is mainly about the time consumption and financial cost of processing the PDTs. In Section III, we introduce the NSGA-II and present an improved algorithm based on it, we apply both of the algorithms to the PDTs scheduling. Section IV compares the experimental results of the two algorithms and

doi:10.4304/jsw.9.2.443-450

evaluates their performance. Section V concludes the paper and discusses some future work.

II. SCHEDULING MODEL

A. Mathematical Model

The PDTs model is an abstract representation of tasks processed in the cloud environment, and we can use directed acyclic graph (DAG) to describe it. For a DAG $G=\{<T>,<E>,<V>\}$, $<T>$ is the set of task nodes, so T_i represents a single task with the subscript i in the DAG, and $amount(T_i)$ represents the calculation amount of T_i. $<E>$ is the set of edges, each edge means a constraint on the two related tasks. For example, E_{ij} is an edge from T_i to T_j, it implies that T_j will never be processed until T_i is completely processed. In addition, $<V>$ is the set of virtual machines associated with a cluster in the cloud computing, and obviously V_i is the virtual machine with the subscript i, $ability(V_i)$ is used to indicate the calculating ability of V_i, and $price(V_i)$ represents the cost of using V_i per unit time. Generally, the greater the $ability(V_i)$ is, the higher the $price(V_i)$ is.

As shown in Fig. 1, the tasks from T_0 to T_7 form a simple workflow while the tasks T_8 and T_9 are independent tasks. All the tasks above and the relations between them compose the PDTs model.

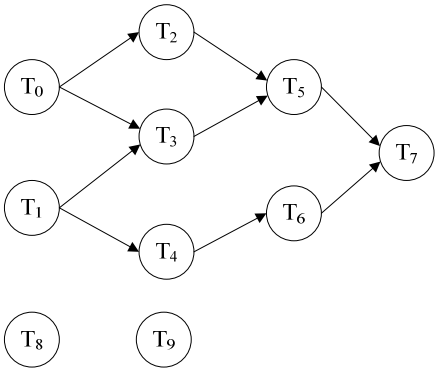

Figure 1. PDTs model

How to allocate the tasks in a PDTs model to the virtual machines in a cluster is an NP-hard problem. In this paper we will take the time consumption and financial cost as the objectives in the multi-objective optimization of PDTs scheduling.

For every task node in the PDTs model, it has to meet the following conditions before it can be processed.

(1) All the direct predecessors of the task node have been completely processed.
(2) The virtual machine that the task is allocated to should be free when the task need to be processed, or the task should have priority over other tasks if they are allocated to the virtual machine at the same time.

Obviously, the constraints between the tasks will affect the finish time of processing all the tasks in the PDTs model, so we should take these constraints into consideration when it comes to the time consumption. We define $Pre(T_i)$ as the collection of direct predecessors of T_i, $Col(T_i)$ as the collection of tasks that conflict with T_i and have priority over T_i in occupying the virtual machine that the tasks are allocated to. We denote the start time of processing T_i by $t_s(T_i)$ and the end time of processing T_i by $t_e(T_i)$. As to $t_s(T_i)$, it depends on $Pre(T_i)$ and $Col(T_i)$, as shown in (1), where $T_k \in Pre(T_i)$, $T_j \in Col(T_i)$, m stands for the number of direct predecessor tasks of T_i and n stands for the number of tasks that conflict with T_i and have priority over it.

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

From (1) we can see that T_i will not be processed until both $Pre(T_i)$ and $Col(T_i)$ are empty, so the time consumption of processing T_i consists of three parts, the first one is the computing time taken by T_i when it is computed by the specified virtual machine, the second one is the waiting time spent by T_i when it waits for the tasks in $Pre(T_i)$ to be completely processed, the last one is the waiting time consumed by T_i when it waits for the tasks in $Col(T_i)$ to be completely processed. So we can describe $t_e(T_i)$ by (2) as follows.

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

The result of $amount(T_i)$ divided by $ability(V_i)$ in (2) stands for the first part of time consumption mentioned above, and the rest parts are described by $t_s(T_i)$ in (1).

B. Fitness Function

This paper focuses on the multi-objective optimization of PDTs scheduling and the fitness function is mainly about the time consumption t_{total} and financial cost c_{total} of processing the PDTs. To simplify the problem of PDTs scheduling, some assumptions are made as follows.

(1) The tasks are non-preemptive.
(2) A task can't be processed by multiple virtual machines simultaneously.

(3) When several tasks are allocated to a virtual machine at the same instant, the task having the highest priority will be processed first.
(4) The greater the *ability*(V_i) is, the higher the *price*(V_i) is.
(5) The tasks are computation-intensive, so we ignore the consumption of network transmission, the memory, and I/O requirements as well.

With all the assumptions above, we can define the time consumption t_{total} by the start time of the first processed task and the end time of the last processed task, as shown in (3).

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

As for the financial cost c_{total}, we define it in (4) where c_i means the cost of using the virtual machine that T_i is allocated to per unit time and p is the number of tasks in the PDTs model.

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

For the users of cloud computing, what they care about is the minimum of time consumption and financial cost, but it's obvious that the two objectives are mutually constrained, so the fitness function is formed by the two factors together, as shown in (5).

$$\begin{cases} \min(t_{total}) = \min(t_e(T_{last}) - t_s(T_{first})) \\ \min(c_{total}) = \min(\sum_{i=1}^{p} c_i * (t_e(T_i) - t_s(T_i))) \end{cases} \tag{5}$$

C. Encoding

One of the most important procedures in the task scheduling of cloud computing is how the tasks are allocated to the virtual machines in the pool of computing resources [14]. For a PDTs scheduling model with p task nodes and q virtual machines, we describe it by a $q \times p$ matrix A where a_{ij} represents the element at the *i*-th row and the *j*-th column of the matrix. We set $a_{ij} = 1$ when T_i is allocated to V_j, otherwise we set $a_{ij} = 0$, $0 \le i \le p-1, 0 \le j \le q-1$. Matrix A has some characteristics as follows. Firstly, it's certain that T_i will be allocated to a virtual machine, so at least one element in the *j*-th column will be assigned to the value of 1. Secondly, according to the assumption (2) in section B of part II, so at most only one element in the *j*-th column can be assigned to the value of 1. Finally, as described in the assumption (3) in the section B of part II, more than one element in the *i*-th row can be assigned to the value of 1.

TABLE1.
ENCODING

T_i / V_j	T_0	T_1	T_2	…	T_n
V_0	0	1	0	…	1
V_1	1	0	0	…	0
V_2	0	0	0	…	0
⋮	⋮	⋮	⋮		⋮
V_j	0	0	1	…	0

Table1 shows an example of encoding which meets all the characteristics mentioned above. The matrix in the table will be used as the encoding of a chromosome involving in the following operators.

III. IMPROVED NSGA-II ALGORITHM

It will be much better for the cloud computing users if the time consumption is shorter meanwhile the financial cost is lower. But we can see from the assumption (4) in the section B of part II that the time consumption and the financial cost are mutually constrained. In other words, the longer time consumption will lead to the lower financial cost. The time consumption and the financial cost can't achieve the minimum at the same time. In this paper, we will take the time consumption and the financial cost as the objectives and use NSGA-II to find the Pareto-optimal front of the PDTs scheduling problem.

As for the Pareto-optimal front, we should make it close to the real optimal solution set as much as possible, and as for the individuals in the Pareto-optimal front, we should make sure that they are distributed as evenly as possible. The traditional NSGA-II algorithm can find a Pareto-optimal front close to the real optimal solution set but it converges slow and the individuals in the Pareto-optimal front are not well distributed. For these disadvantages, we propose an improved algorithm based on NSGA-II in the PDTs scheduling problem.

A. Basic Algorithm

We set the encoding of the relationships between the tasks and the virtual machines in the PDTs model introduced in section C of part II as the encoding of the chromosomes in NSGA-II, and the fitness function described by (5) as the fitness function of the chromosomes in NSGA-II.

The basic steps of NSGA-II are as follows.

Step1: Set the size of the population as M, the maximum evolution generation as maxGen and the current generation index as gen, initialize the population P(gen).

Step2: Choose the chromosomes in P(gen) to perform crossover and mutation operation and generate the new population Q(gen).

Step3: Merge P(gen) with Q(gen) to form R(gen), and do the fast non-dominated sorting in R(gen), the chromosomes are divided into several ranks.

Step4: Calculate the crowding distance for the chromosomes in each rank and sort them according to the ascending order of the crowding distance.

Step5: Select the top M chromosomes in R(gen) into P(gen+1) according to the rank and crowding distance of the chromosomes in R(gen).

Step6: Set gen=gen+1, if gen<maxGen, go to Step2, otherwise go to Step7.

Step7: Take the chromosomes in P(gen) as the individuals in Pareto-optimal front.

B. Improved Strategies

As mentioned above, the disadvantages of the traditional NSGA-II when it is applied to the PDTs scheduling problem are caused by different reasons. We will analyze the reasons and propose improved strategies in the following parts of this paper.

The PDTs scheduling in cloud computing is NP-hard, and the relations between the tasks in the PDTs model are so complex that the traditional NSGA-II algorithm converges slow when it is applied to the problem. Actually, the NSGA-II algorithm tries to find the Pareto-optimal front by retaining the genes of high quality in the chromosomes which will go through the evolution by being chosen to take part in the crossover operation and the mutation operation. Generally, the chromosomes are chosen with certain probability to take part in the single-point crossover and single-point mutation, but these strategies can't retain the excellent genetic fragments to the maximum extent so that the algorithm converges slow. We propose the strategy named STOX to improve the performance of the crossover operator in NSGA-II. STOX can make the algorithm converge faster, but it may lead the algorithm to converging to the local optimal solution. To avoid this situation, we will use the shift mutation strategy.

Another issue needed to be resolved is that the individuals in the Pareto-optimal front are not well distributed. In the PDTs scheduling problem, the time consumption and the financial cost are the two objectives that have different dimensions which causes the enormous value gap between the two objectives, and it will have an adverse effect on computing of the crowding distance because the crowding distance is defined as the sum of the distances between the individual and its two adjacent individuals in the Pareto-optimal front while the distances are determined by the aspects of the two objectives. Therefore we will introduce the self-adapting crowding distance parameter to the computing of the crowding distance to weaken the adverse effect caused by the different dimensions so that the individuals in the Pareto-optimal front are well distributed.

(1) Crossover Strategy

STOX is based on SJOX (Similar Job Order Crossover) [12], it will reserve the similar gene fragments in the chromosomes taking part in the crossover operation so that the offspring can inherit the excellent genetic fragments from the parents and the algorithm can converge faster. However, the encoding of the PDTs scheduling problem is different from the one-dimensional 0-1 encoding, so we propose the STOX which is improved on the basis of the SJOX with corresponding measures for the PDTs scheduling problem. For an illustrative purpose, we will take a PDTs scheduling model with 10 tasks and 3 virtual machines for example as follows.

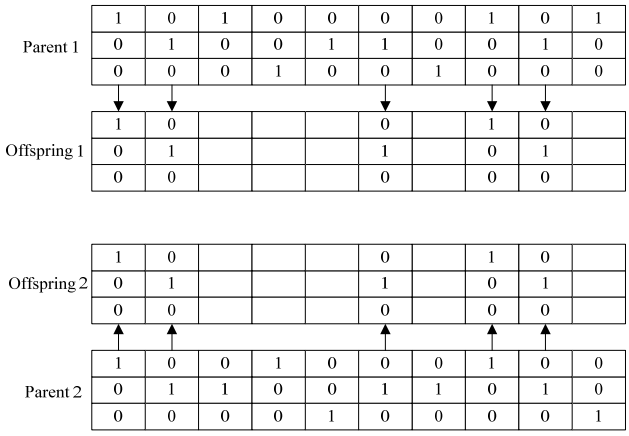

Figure 2. Copy the common tasks from parents to offspring

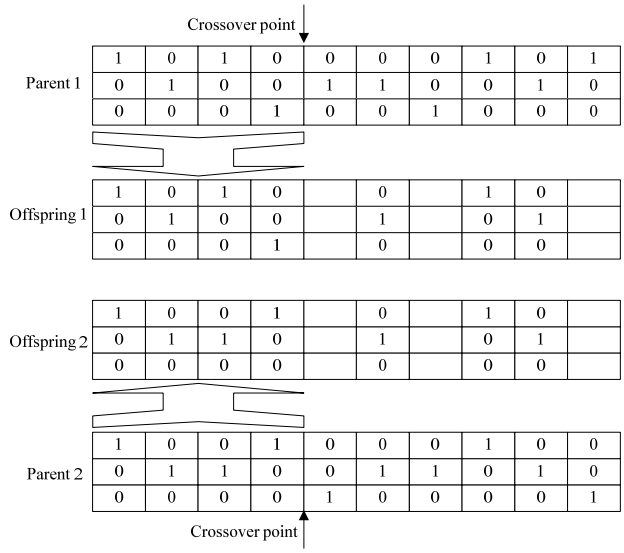

Figure 3. Inherit tasks up to the cut point from the direct parent

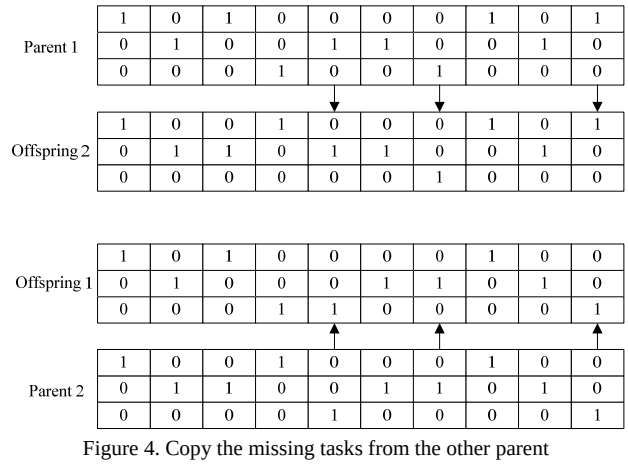

Figure 4. Copy the missing tasks from the other parent

(2) Mutation Strategy

Two column index positions in the coding matrix of the chromosome will be chosen randomly by the shift mutation operator as the starting point and ending point respectively, as for the gene fragments between the starting point and ending point, they will be shift end to end with a step size. We still use the PDTs scheduling model above to illustrate this operator with the step size

set to 1 as follows.

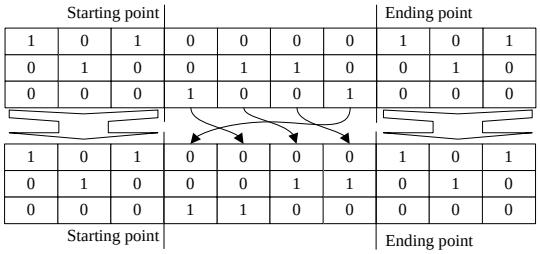

Figure 5. Shift mutation

(3) Self-adapting Crowding Distance Parameter

In the self-adapting crowding distance (SCD) operator, we define the self-adapting crowding distance parameter (SCDP) as the quotient of two operands, one is the maximum gap value of two adjacent individuals in the aspect of the sub-objective time consumption, the other is the maximum gap value of two individuals in the aspect of the sub-objective financial cost. In particular, we'd like to highlight that the boundary values of each rank are not included in the maximum gap values, and we use a variable to count the times that the SCDP doesn't change during the evolution, once the times exceed a certain number such as 5% of the maximum evolution generation, we will believe that the algorithm may converges to the local optimal solution and we should increase the crossover probability and the mutation probability by 0.01 until they reach their respective upper limits. The count variable, as well as the crossover probability and the mutation probability, will be reset if the SCDP changes in the new generation. When we calculate the crowding distance of the individuals at the end of the revolution of each generation, the sub-objective in the denominator of the fraction will be multiplied by the SCDP to weaken the adverse effect caused by the different dimensions of the sub-objectives so that the individuals in the Pareto-optimal front will be well distributed.

Actually, the SCDP is calculated during the Step4 in section A of part III, and it is implemented in the following steps.

Step1: Initialize all the variables including maxCount, maxPc and maxPm, which respectively represent the upper limit of the counter, the crossover probability and the mutation probability. If it is the first generation, set the initial value of the SCDP to 1, otherwise set it with the value of SCDP in the previous generation.

Step2: Calculate the new SCDP in the current generation, if it isn't equal to the SCDP of the previous generation, reset the variables of the counter, the crossover probability and the mutation probability, go to Step6, otherwise, increase the value of the counter by 1.

Step3: If the counter doesn't reach its upper limit, go to Step6, otherwise, go to Step4.

Step4: If the crossover probability reaches its upper limit, go to Step5, otherwise, increase the crossover probability by 0.01.

Step5: If the mutation probability reaches its upper limit, go to Step6, otherwise, increase the mutation probability by 0.01.

Step6: Go outside of the procedure of calculating the SCDP.

The specific steps are shown in Fig. 6.

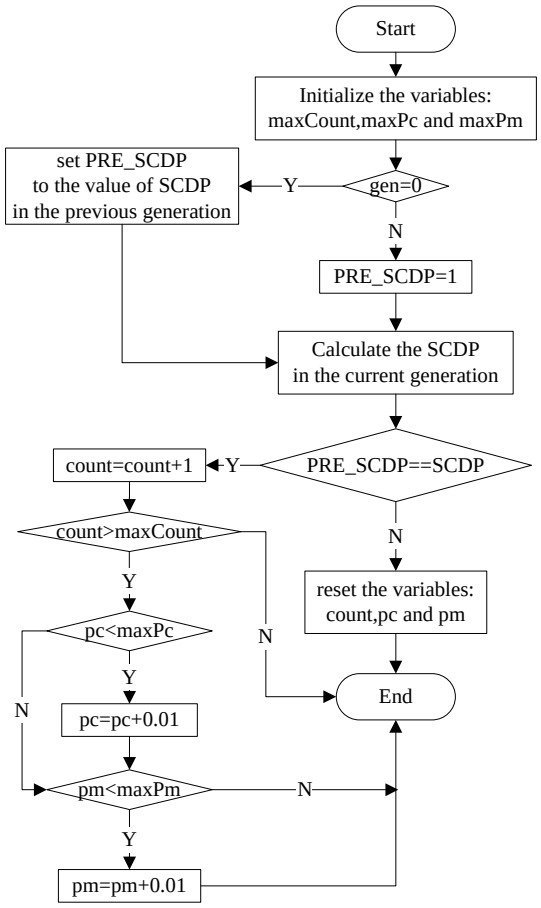

Figure 6. Calculate the SCDP

As mentioned above, the procedure of calculating the SCDP is embedded in the step4 in section A of part III, and the variables such as gen, count, pc and pm are defined outside the procedure.

IV. Experiments And Analysis

In order to evaluate the performance of the improved NSGA-II when applied to the PDTs scheduling problem, we compare it with the traditional NSGA-II in the aspects of the ability of finding the Pareto-optimal solutions and the distribution of the Pareto-optimal solutions in the Pareto-optimal front. With regard to the ability of finding the Pareto-optimal solutions, we take the number of the non-repetitive Pareto-optimal solutions under the same population size and the same evolution generations as reference, obviously, the greater the number is, the more powerful the ability is. As for the distribution of the individuals in the Pareto-optimal front, we take the mean value and the stand deviation of the distances between every two adjacent solutions as reference, the mean value and the stand deviation should be smaller if we expect the individuals in the Pareto-optimal to get a better distribution in the Pareto-optimal front.

A. Performance Comparison

We can compare the performance mentioned above by analyzing the results of the simulation experiments. To get these results, we initialize the scheduling problem by setting a PDTs model with 10 task nodes and 8 virtual machine nodes, and when the algorithms are applied to solve the problem, we set the size of the population ranging from 100 to 500 with an interval value of 100, as well as the evolution generation. For every situation we run the program 50 times and get the average value of the performance mentioned above. Taking the situation with a population of 100 for example, its performance comparisons are shown as follows in Fig.7 and Fig. 8.

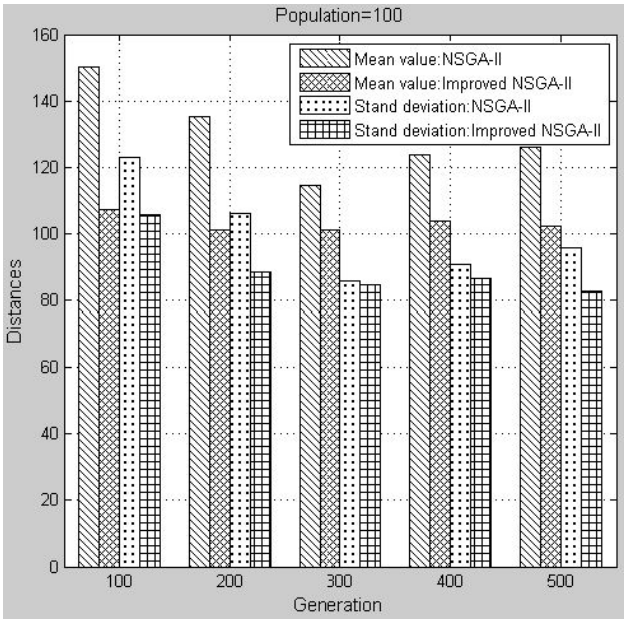

Figure 7.The mean value and the stand deviation of the distances between the adjacent solutions

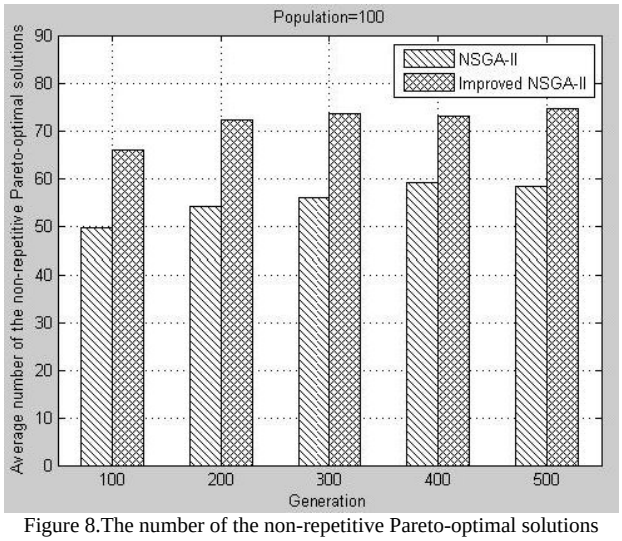

Figure 8.The number of the non-repetitive Pareto-optimal solutions

We can see from the figures that the improved algorithm behaves much better than the traditional algorithm under the same condition, which has a more powerful ability to find the non-repetitive Pareto-optimal solutions meanwhile the solutions in the Pareto-optimal front are better distributed.

Also, we can see that both the improved algorithm and the traditional algorithm behave better as the evolution generation increases when it ranges from 100 to 300, however, when the evolution generation is over 300, the results of the traditional algorithm show that its performance becomes slightly worse while the improved algorithm can avoid the atavism.

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

NSGA-II is essentially an evolutionary algorithm, both the population size and the evolution generation can influence the performance of the algorithm when it is applied to the task scheduling problem. As is shown in the section A of part IV, the improved algorithm behaves much better than the traditional algorithm when they are under the same population size and evolution generation. In this section we take the improved algorithm for example to show that how the algorithm behaves when the evolution generation is sufficient while the population size ranges from 100 to 500 with a step size of 100. Table2 shows the average values of the number of non-repetitive Pareto-optimal solutions, the mean value and the stand deviation of the distances between every two adjacent solutions when the improved algorithm is run 50 times with every different population size.

TABLE2.

INFLUENCE OF POPULATION SIZE ON THE ALGORITHM

Population size	The average number of solutions	Mean value	Stand deviation
100	73.6	103.9076	84.5916
200	123.6	72.4885	68.6235
300	150.2	64.7250	66.0616
400	149.4	65.2252	66.0764
500	141.2	64.7846	68.4530

We can see from the table that the population size has a strong influence on the performance, we should choose an appropriate population size according to the PDTs model's scale and make sure that the evaluation generation is sufficient, as opposed to the supposition that the bigger is the better, for there is a risk of atavism and also a waste of the computing resource and time. So it will be much better if we choose an appropriate population size and evaluation generation or set some limit values to have the computing stop automatically.

C. Pareto-optimal front Comparison

The advantages of the improved NSGA-II are not only reflected in the performance data above but also in the Pareto-optimal front. Actually the improved NSGA-II can get a Pareto-optimal front which is closer to the optimal solution than the traditional NSGA-II when they are under the same population size and evaluation generation. That means the user can get a solution which needs less time consumption or financial cost. Due to the large range in the aspect of financial cost, we only select a subset of the individuals in the Pareto-optimal front to generate a figure for illustrative purpose so that we can see the difference clearly as shown in Fig. 9.

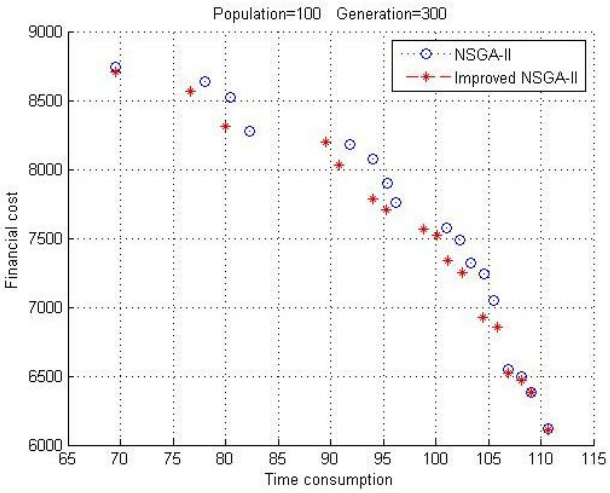

Figure 9.Part of the Pareto-optimal front

V. Conclusion And Future Works

The cloud PDTs are composed of independent tasks and workflow tasks, so the scheduling model is more complicated. In order to find the optimal solutions, we apply the traditional NSGA-II algorithm to this problem and propose an improved algorithm based on it by adopting the STOX, shift mutation and SCD operator. We can see from the simulation experiment that the improved NSGA-II can get a much better Pare-to optimal front than the traditional one in the aspects of the number of non-repetitive Pareto-optimal solutions, the distributions of the individuals in the Pareto-optimal front and the degree of approaching the optimal front when they are under the same population size and evolution generation.

In this paper we are just concerned with the time consumption and financial cost happening in the task computing, however, the data centers of the cloud service providers must consider the bandwidth delay and other issues that may happen in the scheduling procedure. Therefore, the objectives should be more than what we have paid attention to. In the future, we will do further researches on a more complete objective mechanism and simulate with more multi-objective optimization algorithms to check the comparison results.

Acknowledgements

This work was partly supported by National Natural Science Foundation of China (Grand Nos. 41275116) and Jiangsu Economic and Information Technology Commission project of China (Grand Nos.{2011}1178).

References

[1] Armbrust, Michael, et al. "A view of cloud computing." Communications of the ACM 53.4 (2010): 50-58.
[2] Andrzejak, Artur, Derrick Kondo, and Sangho Yi. "Decision model for cloud computing under sla constraints." Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on. IEEE, 2010.
[3] Buyya, Rajkumar, David Abramson, and Jonathan Giddy. "An economy driven resource management architecture for global computational power grids." Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000). 2000.
[4] Vaquero, Luis M., et al. "A break in the clouds: towards a cloud definition." ACM SIGCOMM Computer Communication Review 39.1 (2008): 50-55.
[5] Izakian, Hesam, Ajith Abraham, and Vaclav Snasel. "Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments." Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on. Vol. 1. IEEE, 2009.
[6] Zhu, Hai, et al. "Grid Independent Task Scheduling Multi-Objective Optimization Model and Genetic Algorithm." Journal of Computers 5.12 (2010): 1907-1915.
[7] Pandey, Suraj, et al. "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments." Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010.
[8] Vöckler, Jens-Sönke, et al. "Experiences using cloud computing for a scientific workflow application." Proceedings of the 2nd international workshop on Scientific cloud computing. ACM, 2011.
[9] Shengjun Xue, Jie Zhang, Xiaolong Xu. "An improved algorithm based on ACO for cloud service PDTs scheduling." Advances in Information Sciences and Service Sciences, 2012, 4(18):340-348.
[10] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." Evolutionary Computation, IEEE Transactions on 6.2 (2002): 182-197.
[11] Deb, Kalyanmoy, et al. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II." Lecture notes in computer science 1917 (2000): 849-858.
[12] Ruiz, Rubén, Concepción Maroto, and Javier Alcaraz. "Two new robust genetic algorithms for the flowshop scheduling problem." Omega 34.5 (2006): 461-476.
[13] Murata, Tadahiko, HisaoIshibuchi, and Hideo Tanaka. "Genetic algorithms for flowshop scheduling problems." Computers & Industrial Engineering 30.4 (1996): 1061-1071.
[14] Buyya, Rajkumar, et al. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." Future Generation computer systems 25.6 (2009): 599-616.

Shengjun Xue 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).

Fei Liu was born in Jiangsu, China. He received the B.S degree in 2011 in computer science & technology from Nanjing University of Information Science & Technology, Nanjing, China. He is now studying as a M.S. at Nanjing University of Information Science & Technology, Nanjing, China.

His research interests in cloud storage, cloud scheduling, genetic algorithm and multi-objective optimization.

Xiaolong Xu was born in Jiangsu, China. He received the B.S. degree in 2010 in software engineering from Nanjing University of Information Science & Technology, Nanjing, China. He is now studying as a M.S. at Nanjing University of Information Science & Technology, Nanjing, China. And he will be a Ph.D. student of Nanjing University in September 2013.

His research interests in cloud computing, cloud scheduling and cloud storage. He has 4 papers published on International journals.