International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.4, No.1 (2008), pp. 37–43

A Task Scheduling Algorithm Based on PSO for Grid Computing

Lei Zhang[1], Yuehui Chen[2], Runyuan Sun[1], Shan Jing[1] and Bo Yang[1]

[1]Network Center, University of Jinan
Jiwei Road 106, Jinan, 250022, P.R. China
{zhanglei, , sunry, jingshan, yangbo}@ujn.edu.cn

[2] School of Information Science and Engineering, University of Jinan
Jiwei Road 106, Jinan, 250022, P.R. China
yhchen@ujn.edu.cn

Abstract: **Grid computing is a high performance computing environment to solve larger scale computational demands. Grid computing contains resource management, task scheduling, security problems, information management and so on. Task scheduling is a fundamental issue in achieving high performance in grid computing systems. However, it is a big challenge for efficient scheduling algorithm design and implementation. In this paper, a heuristic approach based on particle swarm optimization algorithm is adopted to solving task scheduling problem in grid environment. Each particle is represented a possible solution, and the position vector is transformed from the continuous variable to the discrete variable. This approach aims to generate an optimal schedule so as to get the minimum completion time while completing the tasks. The results of simulated experiments show that the particle swarm optimization algorithm is able to get the better schedule than genetic algorithm.**

Keywords: **particle swarm optimization, task scheduling, grid computing**

1. Introduction

With the development of the network technology, grid computing used to solve larger scale complex problems becomes a focus technology. Task scheduling is a challenging problem in grid computing environment [1]. If large numbers of tasks are computed on the geographically distributed resources, a reasonable scheduling algorithm must be adopted in order to get the minimum completion time. So task scheduling which is one of NP-Complete problems becomes a focus by many of scholars in grid computing area.

Heuristic optimization algorithm is widely used to solve a variety of NP-complete problems. Abraham et al and Braun et al [2] presented three basic heuristics implied by Nature for Grid scheduling, namely Genetic Algorithm (GA) [3-7], Simulated Annealing (SA) [8-9] and Tabu Search (TS) [10], and heuristics derived by a combination of there three algorithms. GA and SA are powerful stochastic optimization methods, which are inspired form the nature. GA is simulated the evolutionary natural selection process. The better solution of generation is evaluated according to the fitness value and the candidates with better fitness values are used to create further solutions through crossover and mutation processes. Simulated annealing is based on the process of annealing about the solid matter in physics. Both methods are valid and have been applied in various fields due to their strong convergence properties.

Particle Swarm Optimization (PSO) [11] is one of the latest evolutionary optimization techniques inspired by nature. It is simulated the process of a swarm of birds preying. It has the better ability of global searching and has been successfully applied to many areas such as neural network training, control system analysis and design, structural optimization and so on. It also has fewer algorithm parameters than both genetic algorithm and simulated algorithm. Furthermore, PSO algorithm works well on most global optimal problems.

In this paper, PSO algorithm is employed to solve the scheduling problem in a grid environment. Through a serial of simulated experiments, our results show that PSO algorithm is effective for task scheduling in computational grid.

This paper is organized as follows. In Section 2, grid scheduling issues are mainly discussed. Particle swarm optimization algorithm is introduced in section 3. In Section 4, PSO algorithm for task scheduling problem in grid environment is given. Experiment settings and results are

discussed in Section 5 and some conclusions are given in Section 6.

2. Grid scheduling issues

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1]. A resource in the computational grid is something that is required to carry out an operation, for example, a processor used for data processing. The resource management of computational grid is responsible for resource discovery and allocation of a task to a particular resource. Usually it is easy to get the information about the ability to process data of the available resource. In this paper, we will discuss the problem that *n* tasks work on *m* computing resources with an objective of minimizing the completion time and utilizing the resources effectively. If the number of tasks is less than the number of resources in grid environment, the tasks can be allocated on the resources according to the first-come-first-serve rule. If the number of task is more than the number of resources, the allocation of tasks is to be made by some scheduling schemes. Considering the number of tasks is more than the computing resources in this paper, so one task can not be assigned to different resource, implying that the task is not allowed to be migrated between resources[17]. Usually it is able to get the information about the available resources via resource management in the grid environment.

To formulate the problem, define T_i i={1,2,3,…n} as n independent tasks permutation and R_j j={1,2,3,…m} as m computing resources. Suppose that the processing time $\mathrm{P_{i,j}}$ for task i computing on j resource is known. The completion time $\mathrm{C(x)}$ represents the total cost time of completion.

The objective is to find an permutation matrix $x=(x_{i,j})$, with $x_{i,j}$ =1 if resource i performs task j and if otherwise, $x_{i,j}$ =0, which minimizes the total costs

$$C(x)=\sum_{i=1}^{m}\sum_{j=1}^{n}P_{i,j}*x_{i,j} \tag{1}$$

Subject to

$$\sum_{i}^{m}x_{i,j}=1,\forall j\in T \tag{2}$$

$$x_{i,j}\in\{0,1\},\forall i\in R,\forall j\in T \tag{3}$$

The minimal $C(x)$ represents the length of schedule whole tasks working on available resources. The scheduling constraints (2) guarantee that each task is assigned to exactly one resource.

We will discuss that a new optimal schedule is able to find the minimal completion time.

3. Particle swarm optimization algorithm

The particle swarm optimization [12] which is one of the latest evolutionary optimization techniques was introduced in 1995 by Kennedy and Eberhart. PSO algorithm is an adaptive method that can be used to solve optimization problem. Conducting search uses a population of particles. Each particle corresponds to individual in evolutionary algorithm. A flock or swarm of particles is randomly generated initially, each particle's position representing a possible solution point in the problem space. Each particle has an updating position vector X^i and updating velocity vector V^i by moving through the problem space. Kennedy and Eberhart proposed the formula of updating position vector X^i:

$$x_{k+1}^{i}=x_{k}^{i}+v_{k+1}^{i} \tag{4}$$

And the formula of updating velocity vector V^i:

$$\mathrm{v_{k+1}^{i}=w_k v_k^i+c_1 r_1(p_k^i-x_k^i)+c_2 r_2(p_k^g-x_k^i)} \tag{5}$$

Where $\mathrm{c_1}$ and $\mathrm{c_2}$ are positive constant and $\mathrm{r_1}$ and $\mathrm{r_2}$ are uniformly distributed random number in [0,1]. The velocity vector V^i is range of $[-V_{\max},V_{\max}]$ [13].

At each iteration step, a function F^i is calculated by position vector X^i evaluating each particle's quality. The vector P^i represents ever the best position of each particle and P^g represents the best position obtained so far in the population. Changing velocity this way enables the particle to search around its individual best position P^i, and updating global best position P^g, until searing for the global best position in the limited iteration.

4. PSO algorithm for scheduling problem in computational grid

In grid environment, scheduling which is one of a variety of NP-complete problems is a very complicated issue. The aim of this problem is how to improve the efficiency of resource and how to minimize the completion time at the same time. PSO can be implemented to solve various function optimization problems, or some problems which can be transformed to function optimization problems. For solving the task scheduling problem by using PSO algorithm, we use the small position value (SPV) rule [14] which borrowed from the random key representation to

solve the task scheduling problem. The SPV rule can convert the continuous position values to discrete permutation in PSO algorithm.

A population of particles is randomly generated initially. Each particle denoted as X^i (i=1, 2, 3, … , n) with its position, velocity, and fitness value represents a potential solution about resource scheduling. The position of each particle is represented by n number of dimensions as $x_k^i = [x_1^i, x_2^i, \cdots, x_n^i]$ where x_k^i is the position value of i particle with respect to the n dimension, and the velocity is represented by $v_k^i = [v_1^i, v_2^i, \cdots, v_n^i]$ where v_k^i is the velocity of particle i with respect to the n dimension. Based on SPV rules, the continuous position convert to a permutation of sequences S^i, which is a sequence of tasks implied by the particle X^i. S^i is represented by $s_k^i = [s_1^i, s_2^i, \cdots, s_n^i]$, where s_k^i is the sequence of task of i particle in the processing order with respect to the n dimension. Define $r_k^i = [r_1^i, r_2^i, \cdots, r_n^i]$ as the n dimension task processing on the r_k^i resources. The fitness value evaluated by fitness function represents the particle's quality based on the current sequence R^i. The personal best value denoted as P^i represents the best searching position of the particle so far. For each particle in the population, P^i can be determined and updated at each iteration step. In grid resource scheduling with the fitness function $f(R^i)$ where R^i is the corresponding sequence of particle X^i, the personal best P^i of the i particle is obtained such that $f(R_k^i) \le f(R_{k-1}^i)$ where R^i is the corresponding sequence of personal best P^i .For each particle, the personal best is denoted as $p_k^i = [p_1^i, p_2^i, \cdots, p_n^i]$ where p_k^i is the position value of the i particle best with respect to the n dimension. The best particle in the whole swarm is assigned to the global best denoted as G^i. The G^i can be obtained such that $f(R^i) \le f(R_k^i)$, for k = 1, 2, 3, … , m, where R^i is the corresponding sequence of particle best P^i. In general, we define the fitness function of the global best as $f(R_{gbest}^i) = f(R_{best}^i)$, and the global best is defined as $g_k^i = [g_1^i, g_2^i, \cdots, g_n^i]$ where g_k^i is the position value of the global best with respect to the n dimension. Each particle updates its velocity vector based on the experiences of the personal best and the global best in order to update the position of each particle with the velocity currently updated in search space. Each particle keeps track of its own best position and the swarm keeps track of the global best position. In addition, a local search may be applied to a certain group of particles in the swarm to enhance the exploitation of searching space. If the minimum computing time which is able to complete the whole tasks is found or finish with maximum number of iteration, the process will be terminated. The pseudo code of PSO algorithm for task scheduling in grid computing system is given as follows,

BEGIN {
Initialize parameters;
Initialize population randomly;
Initialize each particle position vector and velocity vector;
Find a permutation according to each particle's position;
Evaluate each particle and find the personal best and the global best;
Do {
Update each particle's velocity and position;
Find a permutation according to the updated each particle's position;
Evaluate each particle and update the personal best and the global best;
Apply the local search;
} While (!Stop criterion)
}
END

4.1. Solution representation

For task scheduling algorithm in grid environment, one of the most important issues is how to represent a solution. The solution representation ties up with the PSO algorithm performance. We define one particle as a possible solution in the population. And dimension n corresponding to n tasks, each dimension represents a task. The position vector of each particle makes transformation about the continuous position. We use the smallest position value, namely, the SPV rule is used first to find a permutation corresponding to the continuous position X^i. For the n tasks and m resource problem, each particle represents a reasonable scheduling scheme. The position vector $x_k^i = [x_1^i, x_2^i, \cdots, x_n^i]$ has a continuous set of values. Based on the SPV rule, the continuous position vector can be transformed a dispersed value permutation $s_k^i = [s_1^i, s_2^i, \cdots, s_n^i]$. Then the operation vector $r_k^i = [r_1^i, r_2^i, \cdots, r_n^i]$ is defined by the following formula:

$$R_k^i = S_k^i \bmod m \quad (6)$$

Table 1 illustrates the solution representation of particle X^i of PSO algorithm for 9 tasks and 3 processors. Based on SPV rules, s_k^i is formed to a permutation. Using the formulas (6), $r_{11}^i = s_{11}^i \bmod 3 = 2$ where refers to the sequence number of computing processor. We define that the start number sequence is zero.

Table 1 : Solution Representation

Dimension	x_k^i	s_k^i	r_k^i
0	3.01	5	2
1	7.96	8	0
2	-0.91	0	2
3	0.78	2	2
4	-0.31	1	1
5	1.85	3	0
6	5.26	7	1
7	4.75	6	0
8	1.77	4	1

4.2. Initial Population

The initialized population of particles is constructed randomly for PSO algorithm. The initialized continuous position values and continuous velocities are generated by the follow formula [14];

$$x_k^0 = x_{\min} + (x_{\max} - x_{\min}) * r \quad (7)$$

Where $x_{\min}$ = -0.4, $x_{\max}$ = 4.0 and r is a uniform random number between 0 and 1.

$$v_k^0 = v_{\min} + (v_{\max} - v_{\min}) * r \quad (8)$$

Where $v_{\min}$ = -0.4, $v_{\max}$ = 4.0 and r is a uniform random number between 0 and 1. We think the population size is the number of dimensions. Since the objective is to minimize the completion time, the fitness function value f_k^i is the completion time value which is decoded from the operation repetition vector for particle i.

4.3. The flow of PSO algorithm for task scheduling issues

Initial population randomly of PSO algorithm is the first step of this algorithm. The formulas (4) and (5) are used to construct the initial continuous position values and velocity value of each particle.

The complete flow of the PSO algorithm for the task scheduling of grid can be summarized as follows,

Step1: Initialization.

Set the contents about this algorithm: k_{max} $c_1=c_2=2$ w_0 $iter=1;$

Define the number of active resource and the list of tasks. The dimension of PSO algorithm is the number of tasks.

Initialize position vector and velocity vector of each particle randomly $x_k^0 = [x_1^0, x_2^0, \cdots, x_n^0]$ *and* $v_k^0 = [v_1^0, v_2^0, \cdots, v_n^0]$;

Apply the SPV rule to find the permutation $s_k^0 = [s_1^0, s_2^0, \cdots, s_n^0]$;

Apply the formula (6) to fine the operation vector $r_k^0 = [r_1^0, r_2^0, \cdots, r_n^0]$;

Evaluate each particle I in the swarm using the objective function; find the best fitness value among the whole swarm $f(R^0)$*. Set the global best value* $G^0 = f(R^0)$.

Step 2: Update iteration variable.

$iter = iter + 1;$

Step 3: Update inertia weight.

$\omega^{iter} = \omega^{iter-1} * \beta$;

Step 4: Update velocity.

Table 2 : Local Search Applied to Permutation before Repairing

Dimension	0	1	2	3	4	5	6	7	8
x_k^i	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
s_k^i	5	1	4	3	0	8	2	7	6
r_k^i	2	1	1	0	0	2	2	1	0
x_k^i	1.8	3.01	0.72	-0.45	-2.25	-0.99	5.3	4.8	1.9
s_k^i	4	5	3	2	0	8	1	7	6
r_k^i	1	2	0	2	0	2	1	1	0

Table 3 : Local Search Applied to Permutation after Repairing

Dimension	0	1	2	3	4	5	6	7	8
x_k^i	1.8	-0.99	3.01	0.72	-0.45	-2.25	5.3	4.8	1.9
s_k^i	5	1	4	3	0	8	2	7	6
r_k^i	2	1	1	0	0	2	2	1	0
x_k^i	1.8	0.72	3.01	-0.99	-0.45	-2.25	5.3	4.8	1.9
s_k^i	5	3	4	1	0	8	2	7	6
r_k^i	2	0	1	1	0	2	2	1	0

Table 4 : Parameter settings of PSO and GA algorithm

Algorithm	Parameter description	Parameter Value
PSO	Size of Swarm	30
	Self-recognition coefficient c1	2
	Social coefficient c2	2
	Weight w	0.9 0.4
	Max Velocity	100
GA	Size of population	30
	Probability of crossover	0.8
	Probability of mutation	0.03
	Scale for mutations	0.1

Table 5 : An example of the best result based on PSO algorithm for (3, 10)

Resource	Task									
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
R_1	0	1	1	0	0	1	0	1	0	0
R_2	1	0	0	0	1	0	0	0	1	0
R_3	0	0	0	1	0	0	1	0	0	1

Apply the formula (5) update velocity of each particle;
Step 5: Update position.
Apply the formula (4) update position of each particle;
Step 6: Find permutation.
Apply the SPV rule to find the permutation $s_k^i = [s_1^i, s_2^i, \cdots, s_n^i]$*;*
Step 7: Fine operation vector
Apply the formula (6) to fine the operation vector $r_k^i = [r_1^i, r_2^i, \cdots, r_n^i]$*;*
Step 8: Update personal best.
Each particle is evaluated by using the operation vector to see if the personal best will improve. If $f(R_{best}^i) \le f(R_k^i)$*, then* $f(R_{best}^i) = f(R_k^i)$*;*
Step 9: Update global best.
If $f(R_{gbest}^i) \le f(R_{best}^i)$*, then* $f(R_{gbest}^i) = f(R_{best}^i)$*;*
Step 10: Stopping criterion
If the number of iteration exceeds the maximum number of iteration, then stop; otherwise go to step 2.

4.4. Neighborhood of PSO

In the PSO algorithm, local search is applied to the permutation directly. However, it violates the SPV rule and needs a repair algorithm. The neighborhood structures [14] is showed in Table 2 where $r_2^i = 1$ and $r_4^i = 2$ are interchanged.

As shown in Table 2, applying a local search to the permutation violates the SPV rule because the permutation is a result of the particle's position values. After completing the local search, a particle should be repaired in order to the SPV rule is not violated.

Table 3 shows a new permutation achieved by changing the position values according to the SPV rule after repairing.

The values of positions and permutation are interchanged in terms of their dimensions. When $s_1^i = 1$ and $s_3^i = 3$ are interchanged, their corresponding $x_1^i = -0.99$ and $x_3^i = 0.72$ are interchanged respectively for dimensions k=1 and k=3 to keep the particle consistent with the SPV rule. The advantage of this approach is due to the fact that the repair algorithm is only needed after evaluating all the neighbors in a permutation.

The performance of the local search algorithm depends on the choice of the neighborhood structure. In this paper, the variable neighborhood search (VNS) method presented by Mladenovic and Hansen [18] is adopted. Usually the two neighborhood structures are employed.

S1: Interchange two tasks between a and b dimensions, (a≠b)

S2: Remove the task at the a dimension and insert it in the b dimension, (a≠b)

Where *a* and *b* are the random integer numbers between 1 and the number of tasks. Two swaps and two interchanges are used to diversify the global best solution before applying the local search. The modification is important to direct the search towards the global optima since the global best solution remains the same after some iterations, probably at a local minimum. In addition, neural moves are allowed in the VNS local search in order to restart the search from a different permutation with the same function value.

5. Experimental settings and results

In our experiments we conducted a serial of experiments to test this algorithm on a simulated grid environment. We compared the performance of PSO algorithm with genetic algorithm (GA) [15-16] that have many similarities. Genetic algorithm is an evolutionary natural selection process. The candidate solution of each generation is evaluated according to the high fitness value and is used to create further solutions via crossover and mutation procedures.

The experimental parameter settings of PSO and GA algorithms are described in Table 3. We considered a finite number of processors in our small scale grid environment and assumed that the processing speeds of each processor and the cost time of each task are known. Each experiment was repeated 10 times with different random seeds. We recorded the completion time values of the best solutions throughout the optimization iterations and a minimum cost time of all tasks completed.

In order to analyze the performance of task scheduling algorithm, firstly we had an experiment with a small task scheduling problem. There is 3 resources sand 10 tasks. The speeds of the 3 resources are 4, 3, 2 and the cost time of each task is 19, 23, 24, 20, 20, 25, 16, 21, 24, and 15. The results of GA algorithm running 10 times were {26, 25.4, 25.8, 25.8, 25, 25, 25.8, 26, 25.4, 25}, with an average value of 25.52. The results of PSO algorithm were {25, 25, 25.4, 25.4, 25, 25, 25, 25, 25, 25.2}, with an average value of 25.1. PSO algorithm provided the best result 7 times, while GA algorithm provided the best result 3 times. Table 2 shows an example of the best task scheduling results based on PSO algorithm about (3, 10), which "1" means the task is assigned to the respective resource in grid environment.

We test the task scheduling problem from 5 processors, 100 tasks to 20 processor 200 tasks. In PSO algorithm, the parameters were set that the number of particle is 30, the self-recognition coefficient c_1 and social coefficient c_2 are 2, and the weight is linear decreased from 0.9 to 0.4. For GA, the size of the population is 30.

Figure 1 shows the completion time of PSO and GA about 5 processors and 100 tasks. It displays that PSO usually had better average completion time values than GA. Figure 2 shows three types of test data of running different numbers of tasks. The curves of T1, T2 and T3 denote the results about running 5 processors, 10 processors and 20 processors. Table 6 shows the best result of GA and PSO algorithm about six types of test data. It shows PSO usually spent the shorter time to complete the scheduling than GA algorithm. It is to be noted that PSO usually spent the shorter time to accomplish the various task scheduling tasks and had the better result compared with GA algorithm.

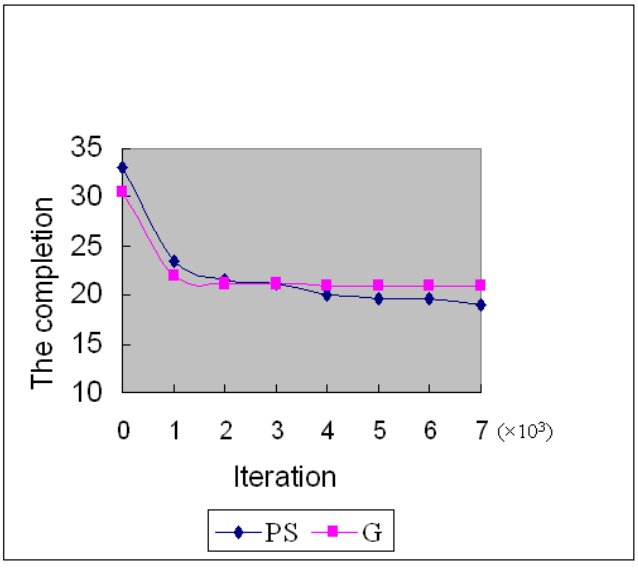

Figure 1 : Performance of PSO and GA scheduling algorithm about 5 processors and 100 tasks.

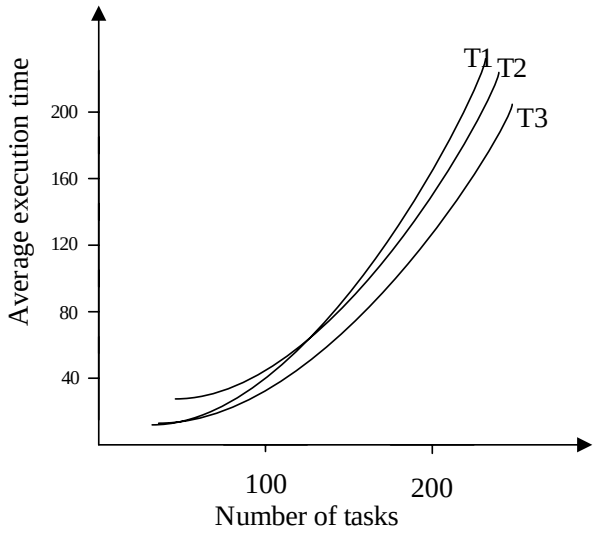

Figure 2 : The performance of curves of different numbers of processors running different numbers of tasks

Table 6 : Performance of GA and PSO algorithm

Problem	GA Algorithm		PSO Algorithm	
	Completion time	Time	Completion time	Time
5*100	208.50	56	198.00	37
5*200	407.00	203	388.50	128
10*100	72.00	78	70.50	39
10*200	190.00	198	183.50	162
20*100	34.00	87	31.50	41
20*200	79.00	277	76.75	158

6. Conclusions

In this paper, scheduling algorithm based on PSO is proposed for task scheduling problem on computational grids. Each particle represents a feasible solution. The position vector is transformed from the continuous values to the discrete values based on SPV rules, accordingly, a permutation formed. Our approach is to generate an optimal schedule so as to complete the tasks in a minimum time as well as utilizing the resources in an efficient way. We evaluate the performance of our proposed approach and compared it with genetic algorithm under the same condition. From the simulated experiment, the result of PSO algorithm is better than GA. Simulation results demonstrate that PSO algorithm can get better effect for a large scale optimization problem. Task scheduling algorithm based on PSO algorithm can be applied in the computational grid environment.

7. References

[1] Foster and C. Kesselman (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufman Publishers, USA, 1999.

[2] Abraham, R. Buyya and B. Nath, Nature's Heuristics for Scheduling Jobs on Computational Grids, The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pp. 45-52,Cochin, India, December 2000,.

[3] Y. Gao, H.Q Rong and J.Z. Huang, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems, pp.1510-161 Elsevier,21(2005).

[4] M. Aggarwal, R.D. Kent and A. Ngom, Genetic Algorithm Based Scheduler for Computational Grids, in Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Application (HPCS'05), ,pp.209-215 Guelph, Ontario Canada, May 2005.

[5] S. Song, Y. Kwok, and K. Hwang, Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling, in Proc. of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), pp.65-74, Denver, Colorado USA, April 2005.

[6] Lee Wang, Howard Jay Siegel, Vwani P. Roychowdhury, and Anthony A. Maciejewski, Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach, Journal of Parallel and Distributed Computing 47, pp.8-22(1997), Foster I. and Kesselman C.

[7] V. D. Martino and M. Mililotti, Sub optimal scheduling in a grid using genetic algorithms, Parallel Computing, pp.553-565, Elsevier,30(2004).

[8] J.E. Orosz and S.H. Jacobson, Analysis of static simulated annealing algorithm, Journal of Optimization theory and Applications, pp. 165-182,Springer,115(1)(2002).

[9] E. Triki, Y. Collette and P.Siarry, A theoretical study on the behavior of simulated annealing leading to a new cooling schedule, pp.77-92, European Journal of Operational Research, Elsevier, 166(2005).

[10] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, pp. 810-837, J. of Parallel and Distributed Computing, vol.61, No. 6,2001.

[11] Kennedy J. and Eberhart R. Swarm Interllignece, Morgan Kaufmann, 2001.

[12] J. Kennedy and R. C. Eberhard, "Particle swarm optimization", Proc. of IEEE Int'l Conf. on Neural Networks, pp.1942-1948, Piscataway, NJ, USA, , 1995.

[13] J.F. Schute and A.A. Groenwold, A study of global optimization using particle swarms, Journal of Global Optimization, pp.93-108, Kluwer Academic Publisher,31(2005).

[14] M. Fatih Tasgetiren, Yun-Chia Liang, Mehmet Sevkli, and Gunes Gencyilmaz, "Particle Swarm Optimization and Differential Evolution for Single Machine Total Weighted Tardiness Problem," International Journal of Production Research, pp. 4737-4754 , vol. 44, no. 22, 2006.

[15] M. Wieczored, R. Prodan and T. Fahringer, Scheduling of Scientific Workflows in the ASKALON Grid Environment, in ACM SIGMOD Record, pp.56-62,Vol.34,No.3, September 2005.

[16] S. Kim and J.B. Weissman, A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications, in Proc. of the 2004 International Conference on Parallel Processing (ICPP'04), pp.406-413, Montreal, Quebec Canada, August 2004.

[17] Lin Jian Ning and Wu Hui Zhong ,Scheduling in Grid Computing Environment Based on Genetic Algorithm, Journal of Computer Research and Development, pp.2195-2199,Vol. 4 ,No.12,Dec 2004.

[18] Mladenovic, N. and P. Hansen. (1997). "Variable Neighborhood Search." Computers and Operations Research, 24, 1097-1100.

[19] Acknowledgment

[20] This research was partially supported by the Natural Science Foundation of China under grant number 60573065, University of Jinan under grant number Y0617 and Y0529X.