# Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm

Ajith Abraham[1,3], Hongbo Liu[2], Weishi Zhang[3], and Tae-Gyu Chang[1]

[1] IITA Professorship Program, School of Computer Science and Engineering, Chung-Ang University, Seoul 156-756, Korea
ajith.abraham@ieee.org
[2] Department of Computer, Dalian University of Technology, 116023, Dalian, China
lhb@dlut.edu.cn
[3] School of Computer Science, Dalian Maritime University,116024, Dalian, China
teesiv@dlmu.edu.cn

**Abstract.** Grid computing is a computing framework to meet the growing computational demands. This paper introduces a novel approach based on Particle Swarm Optimization (PSO) for scheduling jobs on computational grids. The representations of the position and velocity of the particles in the conventional PSO is extended from the real vectors to fuzzy matrices. The proposed approach is to dynamically generate an optimal schedule so as to complete the tasks within a minimum period of time as well as utilizing the resources in an efficient way. We evaluate the performance of the proposed PSO algorithm with Genetic Algorithm (GA) and Simulated Annealing (SA) approaches.

## 1 Introduction

A computational grid is a large scale, heterogeneous collection of autonomous systems, geographically distributed and interconnected by low latency and high bandwidth networks [1]. Job sharing (computational burden) is one of the major difficult tasks in a computational grid environment [2]. Grid resource manager provides the functionality for discovery and publishing of resources as well as scheduling, submission and monitoring of jobs. However, computing resources are geographically distributed under different ownerships each having their own access policy, cost and various constraints. The job scheduling problem is known to be NP-complete. Recently genetic algorithms were introduced to minimize the average completion time of jobs through optimal job allocation on each grid node in application-level scheduling [3],[5]. Because of the intractable nature of the problem and its importance in grid computing, it is desirable to explore other avenues for developing good heuristic algorithms for large scale problems.

Particle Swarm Optimization (PSO) is a population-based optimization tool, which could be implemented and applied easily to solve various function optimization problems, or the problems that can be transformed to function optimization problems [4]. In this paper, fuzzy matrices are used to represent the position and velocity of the particles in the PSO algorithm for mapping the job schedules and the particle. Our approach is to dynamically generate an optimal schedule so as to complete the tasks within a minimum period of time as well as utilizing all the resources.

Rest of the paper is organized as follows. In Section 2, issues related to grid resource management and scheduling is provided following by the proposed PSO based algorithm in Section 3. Experiment results are presented in Section 4 and some conclusions are provided towards the end.

## 2 Grid Resource Management and Scheduling Issues

The grid resource broker is responsible for resource discovery, deciding allocation of a job to a particular grid node, binding of user applications (files), hardware resources, initiate computations, adapt to the changes in grid resources and present the grid to the user as a single, unified resource [5]. To formulate the problem, we consider $J_j$ ($j \in \{1, 2, \cdots, n\}$) independent user jobs on $G_i$ ($i \in \{1, 2, \cdots, m\}$) heterogeneous grid nodes with an objective of minimizing the completion time and utilizing all the computing nodes effectively. The speed of each grid node is expressed in number of Cycles Per Unit Time (CPUT), and the length of each job in number of cycles. Each job $J_j$ has its processing requirement (cycles) and the node $G_i$ has its calculating speed (cycles/second). Any job $J_j$ has to be processed in the one of grid nodes $G_i$, until completion. Since all nodes at each stage are identical and preemptions are not allowed, to define a schedule it suffices to specify the completion time for all tasks comprising each job.

To formulate our objective, define $C_{i,j}$ ($i \in \{1, 2, \cdots, m\}$, $j \in \{1, 2, \cdots, n\}$) as the completion time that the grid node $G_i$ finishes the job $J_j$, $\sum C_i$ represents the time that the grid node $G_i$ completes the processing of all the jobs. Define $C_{max} = max\{\sum C_i\}$ as the makespan, and $\sum_{i=1}^{m}(\sum C_i)$ as the flowtime. An optimal schedule will be the one that optimizes the flowtime and makespan. The conceptually obvious rule to minimize $\sum_{i=1}^{m}(\sum C_i)$ is to schedule Shortest Job on the Fastest Node (SJFN). The simplest rule to minimize $C_{max}$ is to schedule the Longest Job on the Fastest Node (LJFN). Minimizing $\sum_{i=1}^{m}(\sum C_i)$ asks the average job finishes quickly, at the expense of the largest job taking a long time, whereas minimizing $C_{max}$, asks that no job takes too long, at the expense of most jobs taking a long time. Minimization of $C_{max}$ will result in maximization of $\sum_{i=1}^{m}(\sum C_i)$.

## 3 Dynamic Grid Job Scheduling Based on PSO

PSO is a population-based optimization algorithm, which could be implemented and applied easily to solve various function optimization problems, or the problems that can be transformed to function optimization problems [4]. In this Section, we design a fuzzy scheme based on discrete particle swarm optimization [6] to solve the job scheduling problem.

Suppose $G = \{G_1, G_2, \cdots, G_m\}$, and $J = \{J_1, J_2, \cdots, J_n\}$, then the fuzzy scheduling relation from $G$ to $J$ can be expressed as follows:

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \cdots & s_{mn} \end{bmatrix}$$

Here $s_{ij}$ represents the degree of membership of the $i$-th element $G_i$ in domain $G$ and the $j$-th element $J_j$ in domain $J$ with reference to $S$. The fuzzy relation $S$ between $G$ and $J$ has the following meaning: for each element in the matrix $S$, the element

$$s_{ij} = \mu_R(G_i, J_j), i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\}. \tag{1}$$

$\mu_R$ is the membership function, the value of $s_{ij}$ means the degree of membership that the grid node $G_j$ would process the job $J_i$ in the feasible schedule solution. In the grid job scheduling problem, the elements of the solution must satisfy the following conditions:

$$s_{ij} \in [0, 1], i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\}. \tag{2}$$

$$\sum_{i=1}^{m} s_{ij} = 1, i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\}. \tag{3}$$

According to fuzzy matrix representation of the job scheduling problem, the position $X$ and velocity $V$ are re-defined as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}; \qquad V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

The elements in the matrix $X$ above have the same meaning as (1). Accordingly, the elements of the matrix $X$ must satisfy the constraint conditions given in (2), (3). We get the equations (4) and (5) for updating the positions and velocities of the particles based on the matrix operations.

$$V(t+1) = w \otimes V(t) \oplus (c_1 * r_1) \otimes (X^{\#}(t) \ominus X(t)) \oplus (c_2 * r_2) \otimes (X^{*}(t) \ominus X(t)). \tag{4}$$

$$X(t+1) = X(t) \oplus V(t+1). \tag{5}$$

The position matrix may violate the constraints given in (2) and (3) after some iterations, so it is necessary to normalize the position matrix. First we make all the negative elements in the matrix to become zero. If all elements in a column of the matrix are zero, they need be re-evaluated using a series of random numbers within the interval [0,1]and then the matrix undergoes the following transformation without violating the constraints:

$$Xnormal = \begin{bmatrix} x_{11}/\sum_{i=1}^{m} x_{i1} & x_{12}/\sum_{i=1}^{m} x_{i2} & \cdots & x_{1n}/\sum_{i=1}^{m} x_{in} \\ x_{21}/\sum_{i=1}^{m} x_{i1} & x_{22}/\sum_{i=1}^{m} x_{i2} & \cdots & x_{2n}/\sum_{i=1}^{m} x_{in} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}/\sum_{i=1}^{m} x_{i1} & x_{m2}/\sum_{i=1}^{m} x_{i2} & \cdots & x_{mn}/\sum_{i=1}^{m} x_{in} \end{bmatrix}$$

Since the position matrix indicates the potential scheduling solution, we choose the element which has the max value, then tag it as "1", and other numbers in the column are set as "0" in the scheduling array. After all the columns have been processed, we get the scheduling solution from the scheduling array and the makespan (solution). The scheme based on fuzzy discrete PSO for the job scheduling problem is summarized as Algorithm 1, in which the job lists and grid node lists are follows:

– $JList_1$ = Job list maintaining the list of all the jobs to be processed.
– $JList_2$ = Job list maintaining only the list of jobs being scheduled.
– $JList_3$ = Job list maintaining only the list of jobs already allocated ($JList_3 = JList_1 - JList_2$).
– $GList_1$ = List of available grid nodes (including time frame).
– $GList_2$ = List of grid nodes already allocated to jobs.
– $GList_3$ = List of free grid nodes ($GList_3 = GList_1 - GList_2$).

## 4 Experiment Settings, Results and Discussions

In our experiments, Genetic Algorithm (GA) and Simulated Annealing (SA) were used to compare the performance with PSO. Specific parameter settings of all the considered algorithms are described in Table 1. Each experiment (for each algorithm) was repeated 10 times with different random seeds. Each trial had a fixed number of $50 * m * n$ iterations ($m$ is the number of the grid nodes, $n$ is the number of the jobs). The makespan values of the best solutions throughout the optimization run were recorded. And the averages and the standard deviations were calculated from the 10 different trials. In a grid environment, the main emphasis was to generate the schedules as fast as possible. So the completion time for 10 trials were used as one of the criteria to improve their performance.

First we tested a small scale job scheduling problem involving 3 nodes and 13 jobs represented as (3,13). The node speeds of the 3 nodes are 4, 3, 2 CPUT, and the job length of 13 jobs are 6,12,16,20,24,28,30,36,40,42,48,52,60 cycles, respectively. Fig. 1(a) shows the performance of the three algorithms. The results (makespan) for 10 GA runs were {47, 46, 47, 47.3333, 46, 47, 47, 47, 47.3333, 49}, with an average value of 47.1167. The results of 10 SA runs were {46.5, 46.5, 46, 46,46, 46.6667, 47, 47.3333, 47, 47}with an average value of 46.6. The results of 10 PSO runs were {46, 46, 46, 46, 46.5, 46.5, 46.5, 46, 46.5, 46.6667}, with an average value of 46.2667. The optimal result is supposed to be 46. While GA provided the best results twice, SA and PSO provided the best result three and five times respectively. Table 2 shows one of the best job scheduling results for (3,13), in which "1" means the job is scheduled to the respective grid node. Further, we tested the three algorithms for other three $(G, J)$ pairs, i.e. (5, 100), (8, 60) and (10, 50). All the jobs and the nodes were submitted at one time. Fig. 1(b) illustrate the performance curves for the three algorithms during the search process for (10, 50). The average makespan values, the standard deviations and the time for 10 trials are illustrated in Table 3. Although the average makespan value of SA was better than that of GA for (3,13), the case was reversed for

**Algorithm 1.** A scheduling scheme based on fuzzy discrete PSO

```
0     If the grid is active and ($JList_1 = 0$) and no new jobs have been submitted,
      wait for new jobs to be submitted. Otherwise, update $GList_1$ and $JList_1$.
1     If ($GList_1 = 0$), wait until grid nodes are available. If $JList_1 > 0$, update
      $JList_2$. If $JList_2 < GList_1$ allocate the jobs on a first-come-first-serve basis
      and if possible allocate the longest job on the fastest grid node according to the
      LJFN heuristic. If $JList_1 > GList_1$, job allocation is to be made by following
      the fuzzy discrete PSO algorithm detailed below. Take jobs and available grid
      nodes from $JList_2$ and $GList_3$. If $m * n$ ($m$ is the number of the grid nodes, $n$
      is the number of the jobs) is larger than the dimension threshold $D_T$, the jobs
      and the grid nodes are grouped into the fuzzy discrete PSO algorithm loop, and
      the single node flowtime is accumulated. The LJFN-SJFN heuristic is applied
      alternatively after a batch of jobs and nodes are allocated.
2     At $t = 0$, represent the jobs and the nodes using fuzzy matrix.
3     Begin fuzzy discrete PSO Loop
3.0   Initialize the size of the particle swarm $n$ and other parameters.
3.1   Initialize a random position matrix and a random velocity matrix for each par-
      ticle, and then normalize the matrices.
3.2   While (the end criterion is not met) do
3.2.0 $t = t + 1$;
3.2.1 Defuzzify the position, and calculate the makespan and total flowtime for each
      particle (the feasible solution);
3.2.2 $X^* = argmin_{i=1}^{n}(f(X^*(t-1)), f(X_1(t)), f(X_2(t)), \cdots, f(X_i(t)), \cdots, f(X_n(t)))$;
3.2.3 For each particle, $X_i^{\#}(t) = argmin_{i=1}^{n}(f(X_i^{\#}(t-1)), f(X_i(t))$
3.2.4 For each particle, update each element in its position matrix and its velocity
      matrix according to equations (9, 10 and 6);
3.2.5 Normalize the position matrix for each particle;
3.3   End while.
4     End of the fuzzy discrete PSO Loop.
5     Check the feasibility of the generated schedule with respect to grid node avail-
      ability and user specified requirements. Then allocate the jobs to the grid nodes
      and update $JList_2$, $JList_3$, $GList_2$ and $GList_3$. Un-allocated jobs (infeasible
      schedules or grid node non-availability) shall be transferred to $JList_1$ for re-
      scheduling or dealt with separately.
6     Repeat steps 0-5 as long as the grid is active.
```

bigger problem sizes. PSO usually had better average makespan values than the other two algorithms. The makespan results of SA seemed to depend on the initial solutions extremely. Although the best values in the ten trials for SA were not worse than other algorithms, it had larger standard deviations. For SA, there were some “bad” results in the ten trials, so the averages were the largest. In general, for larger $(G, J)$ pairs, the time was much longer. PSO usually spent the least time to allocate all the jobs on the grid node, GA was the second, and SA had to spent more time to complete the scheduling. It is to be noted that PSO usually spent the shortest time to accomplish the various job scheduling tasks and had the best results among all the considered three algorithms.

**Table 1.** Parameter settings for the algorithms

| Algorithm | Parameter name | Parameter value |
|---|---|---|
| GA | Size of the population | 20 |
| | Probability of crossover | 0.8 |
| | Probability of mutation | 0.02 |
| | Scale for mutations | 0.1 |
| SA | Number operations before temperature adjustment | 20 |
| | Number of cycles | 10 |
| | Temperature reduction factor | 0.85 |
| | Vector for control step of length adjustment | 2 |
| | Initial temperature | 50 |
| PSO | Swarm size | 20 |
| | Self-recognition coefficient $c_1$ | 1.49 |
| | Social coefficient $c_2$ | 1.49 |
| | Inertia weight $w$ | $0.9 \rightarrow 0.1$ |

**Table 2.** An optimal schedule for (3,13)

| Grid Node | Job | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ | $J_{10}$ | $J_{11}$ | $J_{12}$ | $J_{13}$ |
| $G_1$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| $G_2$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| $G_3$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**Table 3.** Performance comparison of the three algorithms

| Algorithm | Item | Instance | | | |
|---|---|---|---|---|---|
| | | (3,13) | (5,100) | (8,60) | (10,50) |
| GA | Average makespan | 47.1167 | 85.7431 | 42.9270 | 38.0428 |
| | Standard Deviation | ±0.7700 | ±0.6217 | ±0.4150 | ±0.6613 |
| | Time | 302.9210 | 2415.9 | 2263.0 | 2628.1 |
| SA | Average makespan | 46.6000 | 90.7338 | 55.4594 | 41.7889 |
| | Standard Deviation | ±0.4856 | ±6.3833 | ±2.0605 | ±8.0773 |
| | Time | 332.5000 | 6567.8 | 6094.9 | 6926.4 |
| PSO | Average makespan | 46.2667 | 84.0544 | 41.9489 | 37.6668 |
| | Standard Deviation | ±0.2854 | ±0.5030 | ±0.6944 | ±0.6068 |
| | Time | 106.2030 | 1485.6 | 1521.0 | 1585.7 |

**Table 4.** Run time performance comparison for large dimension problems

| (G,J) | PSO | GA |
|---|---|---|
| (60,100) | 1721.1 | 1880.6 |
| (100,1000) | 3970.80 | 5249.80 |

It is possible that $(G, J)$ is larger than the dimension threshold $D_T$. We considered two large-dimensions of $(G, J)$, (60,500) and (100,1000) by submitting the jobs and the nodes in multi-stages consecutively. In each stage, 10 jobs were allocated to 5 nodes, and the single node flowtime was accumulated. The LJFN-SJFN heuristic was applied alternatively after a batch of jobs and nodes were allocated. Fig. 2 and Table 4 illustrates the performance of GA and PSO during the search process for the considered $(G, J)$ pairs. As evident, even though the performance were close enough, PSO generated the schedules much faster than GA as illustrated in Table 4.

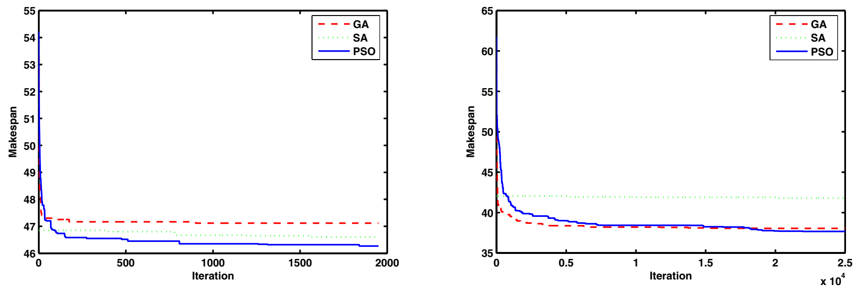

**Fig. 1.** Performance for job scheduling [a] (3,13) and [b] (5,100)

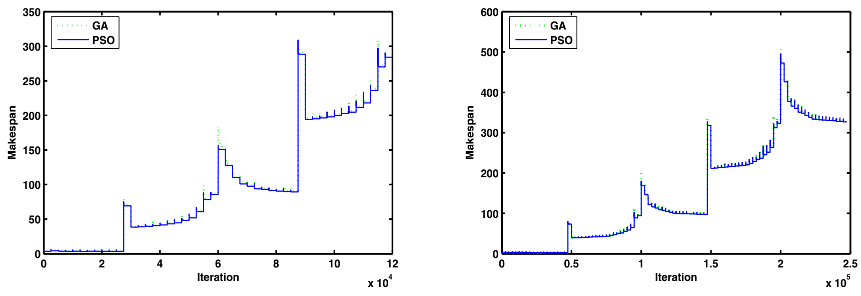

**Fig. 2.** Performance for job scheduling [a] (60,500) and [b] (100,1000) for GA and PSO

## 5 Conclusions

In this paper, we evaluated the performance of a fuzzy particle swarm algorithm for grid job scheduling and compared it performance with genetic algorithms and simulated annealing. Empirical results reveal that the proposed approach can be applied for job scheduling. When compared to GA and SA, an important advantage of the PSO algorithm is its speed of convergence and the ability to obtain faster and feasible schedules.

## Acknowledgements

This research was supported by the International Joint Research Grant of the Institute of Information Technology Assessment foreign professor invitation program of the Ministry of Information and Communication, Korea.

## References

1. Foster,I., Kesselman,C.: The Grid: Blueprint For A New Computing Infrastructure. Morgan Kaufmann, USA (2004).
2. Laforenza,D.: Grid Programming: Some Indications Where We Are Headed Author. Parallel Computing, 28(12) (2002) 1733–1752
3. Gao,Y., Rong,H.Q., Huang,J.Z.: Adaptive Grid Job Scheduling With Genetic Algorithms. Future Generation Computer Systems, 21 (2005) 151–161.
4. Kennedy,J., Eberhart,R.: Swarm Intelligence. Morgan Kaufmann (2001)
5. Abraham,A., Buyya,R., Nath,B.: Nature's Heuristics For Scheduling Jobs on Computational Grids. In: Proceedings of the 8th International Conference on Advanced Computing and Communications, Tata McGraw-Hill, India, (2000) pp. 45-52.
6. Pang,W., Wang,K., Zhou,C., Dong,L.: Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem. In: Proceedings of the Fourth International Conference on Computer and Information Technology, IEEE CS Press (2004) 796–800.