A Modified Binary PSO Algorithm for Scheduling Independent Jobs in Grid Computing System

TARUN KUMAR GHOSH
Department of Computer Science and Engineering, Haldia Institute of Technology, West Bengal, India
SANJOY DAS
Department of Engineering and Technological Studies, Kalyani University, West Bengal, India

Grid computing has been considered as an efficient high performance computing platform to solve large scale and complex computing problems. In computational Grids, the Grid scheduler schedules the submitted jobs and finds the appropriate available resource for each job. Job scheduling which is one of the NP-complete problems has been a focus of many researchers in Grid computing area. This paper presents a modified binary particle swarm optimization (MBPSO) algorithm for efficiently allocating jobs to resources in a Grid system so that makespan and flowtime are minimized. The extensive experimental study shows that our modified binary PSO based scheduler outperforms classical PSO based scheduler and also reveals its efficiency when makespan and flowtime are minimized in varied circumstances.

Keywords: Computational Grid, Job Scheduling, Makespan, Flowtime, PSO.

1. INTRODUCTION

Computational Grid is an emerging computing platform which includes of a large number of heterogeneous computing resources linked by a network across dynamic and geographically dispersed organizations to form a distributed powerful computing paradigm. A computational Grid environment behaves like a virtual organization consisting of distributed resources. A virtual organization consists of a group of individuals and organizations defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing happens [Foster et al., 2001].

Compared to scheduling problems in conventional distributed systems, the job scheduling problem is much more complex as new characteristics of Grid systems such as heterogeneity and dynamic nature of jobs and resources and job's varied demand of resources must be addressed. So, the problem is multi-objective, the two most important objectives being the minimization of makespan and flowtime of the system. Since the Grid resources are dispersed in different geographically regions and belong to various organizational domains, decentralized Grid resource management methods are appropriate solutions. A suitable Grid resource management method utilizes the capability of resources efficiently and satisfies the user's reasonable requests. Job scheduling is a vital and integral part of Grid computing systems. Exhaustive researches in job scheduling have been going on and many results have been widely accepted. Job scheduling is mapping a set of jobs to a set of resources to efficiently utilize the capabilities of them. Optimal mapping of jobs to resources in a Grid computing environment is shown to be NP-complete [Ibarra and Kim, 1977]. Therefore the use of heuristics is the de facto mechanism in order to cope in practice with its difficulty.

In such complex systems, three main stages of scheduling are to be performed [McClatchey et al., 2007]: (a) resource discovery, (b) matchmaking, and (c) job execution. In the resource discovery stage, Grid scheduler performs a global search to prepare a list of all available resources as well as their limitations and history profiles in a system. In the matchmaking stage, scheduler tries to establish best selections for executing jobs. In the job execution stage, scheduler makes commands for selected resources to accomplish jobs.

This paper targets the matchmaking process of schedulers and proposes a modified binary PSO (MBPSO) scheduling algorithm to concurrently minimize two key performance issues of a computational Grid system: (a) makespan for executing all jobs, and (b) flowtime of all jobs. Makespan is the time when Grid finishes the latest job and flowtime is the sum of finalization times of all the jobs. Our experimental results prove that the modified version of PSO algorithm performs better than the classical PSO.

This paper is organized as follows. Section 2 introduces Grid scheduling issues. Section 3 briefly outlines the relevant past works done on job scheduling in computational Grid environment. In Section 4, the framework of Grid job scheduling problem has been defined. Section 5 highlights particle swarm optimization method. Section 6 describes a proposed modified binary particle swarm optimizer for scheduling jobs in computational Grid systems. Section 7 exhibits the results obtained in this study. Finally, Section 8 concludes the paper.

2. GRID SCHEDULING ISSUES

Most Grid systems use the Grid resource broker which is responsible for resource discovery, deciding allotment of a job to a particular resource, linking of user programs to hardware resources, initiating computations, adapting to the changes in Grid resources and presenting the Grid to the user as a single unified resource. It finally controls the physical allotment of the jobs and manages the available resources constantly while dynamically updating the Grid scheduler whenever there is a change in resource availability [Abraham et al., 2000].

In Grid systems, existing scheduling techniques can be categorized into two groups [Macheswaran et al., 1999]: immediate mode (on-line mode) and batch-mode. In the immediate mode, each job is mapped onto an available resource as soon as it enters the system. In the batch mode, a job is not mapped onto a resource immediately; instead, jobs are grouped in batches and scheduled as a group. The immediate mode technique is appropriate for the low arrival rate, while batch mode method can achieve higher performance when the arrival rate of jobs is high because there will be a sufficient number of jobs to keep resources busy between the mapping events, and scheduling is according to the resource requirement information of all jobs in the set. In this paper, we have considered batch mode scheduling.

Grid environments are dynamic by nature. There are basically two main characteristics that decide the dynamics of the Grid scheduling, namely [Xhafa and Abraham, 2010]: (a) The dynamics of job execution, which refers to the situation when job execution could fail or, in the preemptive mode, job execution is stopped due to the arrival of high priority jobs in the system; and (b) The dynamics of resources, in which resources can join or drop the Grid randomly, their workload can drastically vary over time, the local policies on usage of resources could change over time, etc. These two issues decide the behavior of the Grid scheduler, ranging from static to highly dynamic. For example, in the static case, there is no job failure and resources are assumed available all the time (e.g. in Enterprise Grids). Although this is unrealistic for most Grids, it could be useful to consider for batch mode scheduling: the number of jobs and resources is considered fixed during short intervals of time.

It should be noted, however, that meta-heuristics run on static instances of the problem and therefore in our approach (i.e. MBPSO) static schedulers are obtained. In order to deal with the dynamics of the Grid systems, dynamic schedulers run the static PSO scheduler in batch mode to schedule jobs arrived in the system since its last activation. Surely, the efficiency of the heuristic is critical for the superiority of the dynamic scheduler.

3. RELEVANT WORKS

Many works have been done in the area of job scheduling in computational Grids. Heuristic optimization methods can be applied to resolve such complex problems [Schopf, 2004]. On the other hands, meta-heuristic algorithms are extensively used to solve a variety of NP-hard problems.

Commonly, there are three categories of heuristic methods used for job scheduling in Grid environment [Xhafa and Abraham, 2010]: (a) Local search based heuristic approach, in which the solution space is explored by jumping from one solution to another one and constructing thus a path in solution space with the aim of finding the best solution for the problem. Examples in this family include Hill Climbing (HC), Simulated Annealing (SA) and Tabu Search (TS) method. (b) Population based heuristic approach which maintains and improves multiple candidate solutions often using population characteristics to guide the search. Such family of methods usually requires long running times if sub-optimal or optimal solutions are to be found. Mainly the following population based heuristic methods are used in job scheduling in Grid: Genetic Algorithms (GAs), Memetic Algorithms (MAs), Ant Colony Optimization (ACO), Cuckoo Search (CS) and Particle Swarm Optimization (PSO). (c) Hybrid heuristic approach which combines elements of several pure heuristics approaches. Meta-heuristic methods are per se hybrid approaches. For instance, GA is combined with local search heuristics such as TS and SA [Abraham et al., 2000]. However, hybridization among different meta-heuristics has been shown to be efficient for many problems by outperforming single methods [Talbi, 2002].

Hill Climbing (HC) method can produce a feasible solution of certain quality within a very short time. This method has been applied for the scheduling under the ETC model by Ritchie and Levine [2003]. SA is more powerful than simple local search by accepting also poorer solutions with certain probability. Such heuristic has been studied for Grid scheduling by Abraham et al. [2000], Goswami et al. [2011] and Yarkhan and Dongarra [2002]. Tabu Search (TS) is more sophisticated and generally requires longer computation time to achieve good solutions. However, its methods of tabu lists, aspiration criteria, intensification and diversification make it a very powerful search algorithm. Abraham et al. [2000] have applied TS for the problem. Ritchie [2003] has proposed TS for the problem under the ETC model and used it in combination with an ACO method.

GAs for Grid scheduling problems have been studied by Abraham et al. [2000], Braun et al. [2001], Zomaya and Teh [2001], Martino and Mililotti [2004], Page and Naughton [2005], Gao et al. [2005], Xhafa and Duran et al. [2008] and Aggarwal and Kent [2005]. Memetic Algorithms (MAs) [Moscato, 1989] is a comparatively new category of population based methods which use both the ideas of evolutionary search and local search. In other words, MAs can be considered as hybrid evolutionary algorithms. Xhafa [2007] has used unstructured MAs and Xhafa and Alba et al. [2008] have proposed Cellular MAs (structured MAs) for independent scheduling under the ETC model. An ACO implementation for the scheduling problem under the ETC model has been investigated by Ritchie [2003]. Lorpunmanee et al. [2007] have also applied an ACO method for dynamic job scheduling in Grid environment. Prakash et al. [2012] and Rabiee and Sajedi [2013] have proposed job scheduling in Grid using Cuckoo Search (CS). Zhang et al. [2008] have proposed a heuristic approach based on PSO algorithm to solving job scheduling problem in Grid environment and it shows that PSO algorithm has produced better results compared to that using GA. An approach for scheduling using a fuzzy PSO algorithm has been proposed by Liu et al. [2010].

4. PROBLEM DEFINITION

The Grid resource broker is responsible for scheduling the jobs received from the users and querying their required services in Grid information services and then assigning these jobs to the discovered services (resources). In this paper, the model of Ali et al. [2000] is used for simulating job-resource mapping process in computational Grid environment. Using this model will allow us to make realistic simulations of the Grid environment.

To formulate the problem we assume that in a specific time interval users have submitted n jobs $\{J_1, J_2, ..., J_n\}$ to the Grid resource broker. Also assume the jobs are independent of each other, each job has to be processed in the one of Grid resources until completion and preemption is not allowed. Consider that there are m resources $\{R_1, R_2, ..., R_m\}$ are available in the Grid system at

the time of receiving the jobs by Grid resource broker. In this paper scheduling is performed at the resource level and it is assumed that each resource follows First-Come First-Served (FCFS) scheme for executing the received jobs.

In this paper, our main goal of the job scheduling problem in computational Grid systems is to efficiently allocate independent jobs to resources so that two important parameters: makespan and flowtime get minimized. To formulate our goal, the ETC (*Expected Time to Compute*) matrix model is used. An ETC matrix is an n x m matrix in which n is the number of jobs and m is the number of resources. It is assumed that an accurate estimation of the computational needs (measured in millions of instructions) of each job and the computing capacity (measured in million instructions per second (MIPS)) of each resource are known prior to execution and contained within an ETC matrix. Each row of the ETC matrix specifies the estimated execution time for a given job on each resource. Similarly, each column of the ETC matrix indicates the estimated execution time of a given resource for each job. $ETC[i, j]$ is the expected execution time of job i on the resource j. The previous workload of a resource is measured by the time $readyj$ when the machine j will have finished the previously assigned jobs. The size of the ETC matrix is no_of_jobs x $no\_of\_resources$. For the simulation studies, characteristics of the ETC matrices were varied in an attempt to represent a range of possible heterogeneous environments. There are several parameters for measuring the quality of solutions for the problem of scheduling jobs in computational Grids. Thus, this problem is a multi-objective in its general formulation.

In this paper, the simultaneous minimization of makespan and flowtime are considered. The makespan is the time when finishes the latest job and the flowtime is minimizing the sum of finalization times of all the jobs. These two criteria are defined as follows:

$$Minimization\ of\ makespan = \min_{S_i \in Schd} [\max_{j \in Jobs} F_j] \tag{1}$$

$$Minimization\ of\ flowtime = \min_{S_i \in Schd} [\sum_{j \in Jobs} F_j] \tag{2}$$

where F_j denotes the time when job j finalizes, $Schd$ is the set of all possible schedules and $Jobs$ is the set of all jobs to be scheduled. Note that makespan is not affected by any particular execution sequence of the jobs in a concrete resource; while in order to minimize flowtime of a resource, jobs should be executed in an increasing order of their expected time to compute.

5. PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization (PSO) is a population based optimization technique inspired by bird flocking and fish schooling originally designed and introduced by Kennedy and Eberhart [1995]. A PSO algorithm considers a swarm of particles in which each particle corresponds to a potential solution. A swarm is similar to a population while a particle is similar to an individual. The particles fly through a multidimensional search space in which the position of each particle is determined based on its own experience and the experience of its neighbors. PSO algorithm combines local search methods (through self experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation [Salman et al., 2002].

PSO algorithm consists of a population of particles whose positions represent the potential solutions for the given problem, and velocities are randomly initialized in the search space. In each iteration, the search for optimal position is performed by updating the particle velocities and positions. Each particle's position will have a fitness value, which will be evaluated by a fitness function to be optimized in each generation. The velocity of each particle is updated using two best positions, personal best position and neighborhood best position. The personal best position, which is denoted by *pbest*, is the best position the particle has visited. The neighborhood best position, which is denoted by *nbest*, is the best position the particle and its neighbors have visited

ever. Two PSO algorithms can be developed according to the size of neighborhoods. When all of the population size of the swarm is considered as the neighbor of a particle, *nbest* is called global best and is denoted by *gbest*. On the other hand, if the smaller neighborhoods are considered for each particle, then *nbest* is called local best and is denoted by *lbest*. *gbest* uses the star neighborhood topology and *lbest* usually uses ring neighborhood topology. There are two main differences between *gbest* and *lbest* with respect to their convergence characteristics [Eberhart et al., 1996]. Due to the larger particle interconnectivity of the *gbest* PSO it converges faster than the *lbest* PSO, but *lbest* PSO is less vulnerable to being trapped in local optima. The velocity and position of each particle are updated as follows:

$$V_k(t+1) = \omega V_k(t) + c_1 r_1 (pbest_k - X_k(t)) + c_2 r_2 (nbest_k - X_k(t)) \tag{3}$$

$$X_k(t+1) = X_k(t) + V_k(t+1) \tag{4}$$

where c_1 and c_2 are acceleration constants which control the influence of *pbest* and *nbest* on the search process. r_1 and r_2 are random numbers in the interval [0, 1]. $V_k(t)$ and $X_k(t)$ represent the velocity and position of the kth particle at iteration t respectively. $pbest_k$ and $nbest_k$ are the best values of positions which are achieved respectively for the kth particle and all particles so far. ω is the inertia weight which regulates the impact of the previous velocity value on the current velocity value. It is observed that a larger value of inertia weight supports global exploration while a small one encourages local exploration. Appropriate choice of inertia weight ω usually gives a balance between the global and local exploration and decreases the average number of iterations to trace the optimum solution. To reach a high performance, we linearly reduce the value of inertia weight from about 0.9 to 0.4 during a run.

6. MODIFIED BINARY PSO (MBPSO)

Mainly the PSO has been used as search-technique in continuous spaces. However, there are many real world optimization problems which involve a set of discrete variables cannot be solved by the standard PSO method. Aiming at this large type of problems, Kennedy and Eberhart [1997] developed a binary version of PSO. In line of this binary PSO, a modified binary PSO (MBPSO) can be proposed. Velocities of the particles are defined in terms of probabilities that a bit will be in one state or the other. The position of a particle can take the binary value of 0 or 1 in which 1 means "included" and 0 means "not included". Thus, velocity and position of each particle can change from 0 to 1 and vice versa. The position update equation is modified as

$$X_k(t+1) = \begin{cases} 1, & \text{if } sig(V_k(t+1)) > rand() \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $rand()$ is a uniform random number in the range [0, 1]. $sig(V_k(t+1))$ is the sigmoid function given by

$$sig(V_k(t+1)) = \frac{1}{1 + exp(-V_k(t+1))} \tag{6}$$

However, the velocity update equation remains same as Eq. (3).

6.1 MBPSO ENCODING FOR JOB SCHEDULING

For a successful PSO algorithm design, the representation step is very important. That means finding a suitable mapping between problem solution and PSO particle is a key issue. This paper encodes each particle's position in an n-dimensional search space in which n is the number of jobs to be scheduled. The value of each dimension is represented by binary numbers included in range [1, m] signifying the machine/resource number, in which m is the number of available resources

in Grid system at the time of scheduling. Assume that $X_k = \{X_{k1}, X_{k2}, ..., X_{kn}\}$ represents the position of kth particle; X_{kj} denotes the machine/resource where job J_j is assigned by the scheduler in this particle. It can be noted that in this encoding scheme a resource number can be seen more than once in a particle.

Since two positions, *pbest* and *nbest*, indicate the personal best position and neighborhood best position of each particle respectively, so the *pbest* and *nbest* encoding is similar to the particle's position. Also this paper follows star neighborhood topology for *nbest*. It means *gbest* PSO. In our modified binary PSO technique, each particle's velocity is assumed as an m x n binary matrix.

For updating particle's position we use the updated velocity matrix and a heuristic, η which adds an explicit bias towards the most attractive solutions and is a problem-dependent function. In our proposed method for updating a particle's position, for each job, the probability of its performing on various machines is calculated as

$$p_{kij} = \frac{V_{kij} \times [\eta_{kij}]^{\beta}}{\sum_{l=1,2,....,m} V_{klj} \times [\eta_{klj}]^{\beta}} \tag{7}$$

Here p_{kij} is the probability of performing job J_j on machine M_i in particle k, and η_{kij} represents a priori effectiveness of performing job J_j on machine M_i in particle k. The pseudo code of the MBPSO algorithm is given in Algorithm: 1.

Algorithm: 1 - Modified Binary PSO (MBPSO) Algorithm

Create and initialize swarm with P particles which are randomly chosen from binary values 0 and 1.
// X, *pbest*, *nbest* are n-dimensional and V is m x n matrix.
While terminating condition is not met
For each particle k = 1,2,....,P
If $f(X_k) > f(pbest_k)$ // f() is the fitness function
$pbest_k = X_k$;
EndIf
If $f(pbest_k) > f(nbest_k)$
$nbest_k = pbest_k$;
EndIf
EndFor
For each particle k = 1,2,...,P
For each job j = 1,2,....,n
If $X_{kj} \neq pbest_{kj}$
$V_{k(X_{kj})j} = V_{k(X_{kj})j}$ - $c_1 r_1$;
$V_{k(pbest_{kj})}j = V_{k(pbest_{kj})j} + c_1 r_1$;
EndIf
If $X_{kj} \neq nbest_{kj}$
$V_{k(X_{kj})j} = V_{k(X_{kj})j}$ - $c_2 r_2$;
$V_{k(nbest_{kj})j} = V_{k(nbest_{kj})j} + c_2 r_2$;
EndIf
EndFor
For each job j = 1,2,...,n
For each machine i = 1,2,....,m
calculate p_{kij} using Eq. (7);
EndFor
select a machine having maximum p_{kij} for allocating to job J_j;
update the workload of the selected machine;
EndFor
EndFor
EndWhile

7. EXPERIMENTS & RESULTS

In order to evaluate the performance of proposed MBPSO algorithm, it is compared with standard PSO algorithm for the job scheduling problem in computational Grids. The objective of scheduler in both algorithms is to minimize the makespan and flowtime. Our tests are performed on a Pentium IV 2.6 GHz, 4GB RAM, and the algorithms are implemented using MATLAB Release 2013A. To enhance the performance of the proposed algorithm and standard PSO, a fine change is made and best values for their parameters are selected which are given in Table I. Also the benchmark by Ali et al. [2000] is used for realistic simulations in this paper.

Table I: Parameter settings for the algorithms

Algorithm	**Parameter name**	**Parameter value**
PSO	Population size	25
	Maximum number of iterations	500
	$c_1 = c_2$	1.49
	$r_1 = r_2$	0.8
	Intertia weight (ω)	$0.9 \rightarrow 0.4$
MBPSO	Population size	25
	Maximum number of iterations	500
	$c_1 = c_2$	1.49
	$r_1 = r_2$	0.8
	Intertia weight (ω)	$0.9 \rightarrow 0.4$

In the simulation model proposed by Ali et al. [2000], expected time to compute (ETC) matrix is used for 512 jobs and 16 resources. The instances of this benchmark model are divided into 12 different groups based on three metrics: job heterogeneity, resource heterogeneity and consistency. They are labelled as *u-x-yy-zz* where u means uniform distribution in the ETC matrix generation, x means the type of consistency (c stands for consistent, i for inconsistent and s for semi-consistent), yy and zz imply the job and resource heterogeneity (hi stands for high, and lo for low). An ETC matrix is said to be consistent when if a resource is faster than other for some job, then it is faster for all the jobs. The matrix is inconsistency when if a resource is faster for some jobs and slower for some others, while it is semi-consistent when if it contains a consistent sub-matrix.

The makespans and flowtimes obtained using standard PSO and MBPSO are compared in Tables II and III respectively. The results are averaged over 10 independent runs. In Tables II and III, the first column indicates the instance name, the second and third columns indicate the mean makespan and flowtime (in second) obtained by PSO and MBPSO respectively. Table II shows the minimum average makespan for the proposed MBPSO algorithm in comparison with the classical PSO algorithm for all 12 instances of the problem. Like-wise, Table III indicates the minimum average flowtime for the proposed MBPSO algorithm in comparison with the classical PSO algorithm for all 12 instances. The statistical results of two algorithms in terms of the mean makespan and flowtime for the 12 instances are summarized in Figures 1 and 2 respectively. From the figures, it can be seen that our proposed MBPSO algorithm produces reduced mean makespan and flowtime values. Therefore, the proposed MBPSO algorithm outperforms the classical PSO algorithm for the problem in hand.

8. CONCLUSIONS

This paper presents a modified binary PSO (MBPSO) implementation for scheduling independent jobs in Grid systems. This scheduling problem has been receiving much attention from researchers due to its importance in obtaining high performance out of complex heterogeneous Grid systems. PSO has been considered here to deal with the complexity of the problem and because it has shown to be very successful for a wide range of optimization problems, including scheduling problems. This is to note that the PSO was earlier proposed for solving the scheduling problem by Zhang

Table II: Comparison of statistical results for average makespan (in second) obtained by PSO and MBPSO algorithms

Instance	PSO	MBPSO
u-c-hi-hi	14660686	**13461645**
u-c-hi-lo	225412	**223425**
u-c-lo-hi	464149	**452246**
u-c-lo-lo	7567	**7502**
u-i-hi-hi	23131142	**22021355**
u-i-hi-lo	288175	**282273**
u-i-lo-hi	849816	**834834**
u-i-lo-lo	9454	**9409**
u-p-hi-hi	22168750	**21238423**
u-p-hi-lo	268834	**262322**
u-p-lo-hi	765853	**753422**
u-p-lo-lo	8741	**8692**

Table III: Comparison of statistical results for average flowtime (in second) obtained by PSO and MBPSO algorithms

Instance	PSO	MBPSO
u-c-hi-hi	198942721	**174532423**
u-c-hi-lo	3325755	**3303552**
u-c-lo-hi	7045483	**6835432**
u-c-lo-lo	115798	**112354**
u-i-hi-hi	342389562	**323279232**
u-i-hi-lo	4340281	**4311137**
u-i-lo-hi	12525844	**11238251**
u-i-lo-lo	141074	**140328**
u-p-hi-hi	324146576	**308342874**
u-p-hi-lo	3939173	**3903274**
u-p-lo-hi	11368754	**11152551**
u-p-lo-lo	126175	**121625**

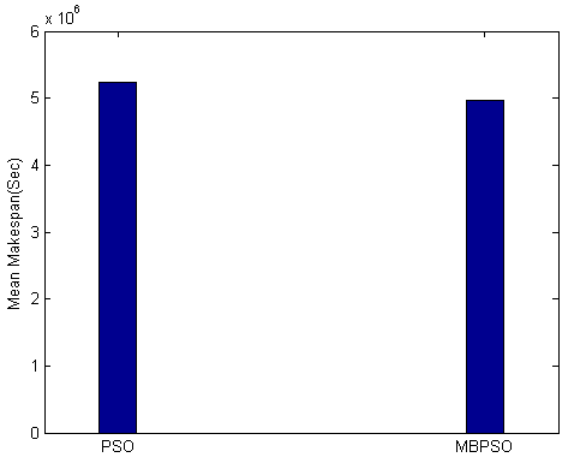

Figure 1: Mean makepan comparison between standard PSO and MBPSO algorithms

et al. [2008], but the reported performance results are not encouraging for a Grid system given its dynamic nature. Therefore, our main objective was to obtain an efficient implementation that would yield to a scheduler for minimizing makespan and flowtime as a multi-objective problem

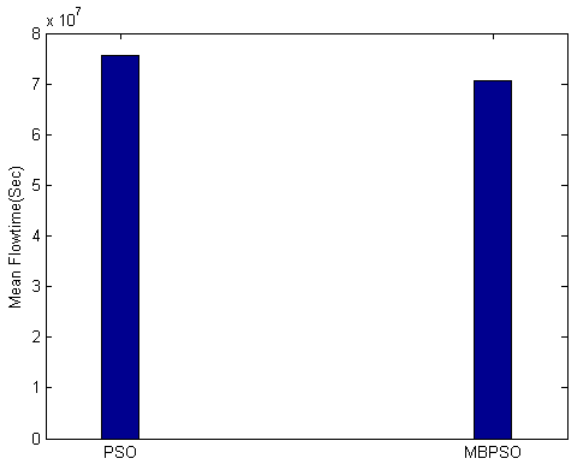

Figure 2: Mean flowtime comparison between standard PSO and MBPSO algorithms

in Grid systems. Our computational results show that our MBPSO scheduler outperforms the standard PSO implementation for all the considered instances. The future research will be directed towards other QoS parameters such as CPU utilization, communication delay and so on.

REFERENCES

Abraham, A., Buyya, R., and Nath, B. 2000. Nature's heuristics for scheduling jobs on computational grids. In *Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, (ADCOM 2000), Tata McGraw-Hill Publishing Co. Ltd, New Delhi, 45-52.*

Aggarwal, M., and Kent, R. 2005. Genetic Algorithm Based Scheduler for Computational Grids. In *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS '05).*

Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D., and Ali, S. 2000. Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems. *Tamkang Journal of Science and Engineering. 3(3), 195-207.*

Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., and Yao, B. 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *International Journal of Parallel and Distributed Computing, 61(6), 810-837.*

Eberhart, R.C., Simpson, P.K., and Dobbins, R.W. 1996. Computational Intelligence PC Tools. *Academic Press Professional, San Diego.*

Foster, C., Kesselman, C., and Tuecke, S. 2001. The anatomy of the Grid. *International Journal of Supercomputer Applications, 15(3), 200-222.*

Gao, Y., Rong, H., and Huang, J.Z. 2005. Adaptive Grid job scheduling with genetic algorithms. *Future Generation Computer Systems, Elsevier, 21(1), 151-161.*

Goswami, R., Ghosh, T. K., and Barman, S. 2011. Local search based approach in Grid scheduling using simulated annealing. In *Proceedings of IEEE International Conference on Computer and Communication Technology (ICCCT-2011), Allahabad, India.*

Ibarra, O.H., and Kim, C.E. 1977. Heuristic algorithms for scheduling independent tasks on non-identical processors. *Journal of ACM, 24(2), 280-289.*

Kennedy, J., and Eberhart, R.C. 1995. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks, 1942-1948.*

Kennedy, J., and Eberhart, R.C. 1997. A discrete binary version of the particle swarm algorithm. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL, 5, 4104 - 4108*

Liu, H., Abraham, A., and Hassanien, A.E. 2010. Scheduling jobs on computational Grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems, Elsevier, 26(8), 1336-1343.*

Lorpunmanee, S., Sap, M.N., Abdullah, A.H., and Inwai, C.C. 2007. An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment. *International Journal of Computer Electrical, Automation, Control and Information Engineering, 1(5), 1343-1350.*

Macheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., and Freund, R.F. 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel Distributed Computing, 59(2), 107-131.*

Martino, V.D., and Mililotti, M. 2004. Sub optimal scheduling in a grid using genetic algorithms. *Journal of Parallel Computing, 30, 553-565.*

McClatchey, R., Anjum A., Stockinger, H., Ali, A., Willers, I., and Thomas, M. 2007. Data intensive and network aware (DIANA) grid scheduling. *Journal of Grid Computing, 5, 43-64.*

Moscato, P. 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In *CalTech Concurrent Computation Program, California Institute of Technology, USA, Technical Report No. 826.*

Page, J., and Naughton, J. 2005. Framework for task scheduling in heterogeneous distributed computing using genetic algorithms. *AI Review, 24, 415-429.*

Prakash, M., Saranya, R., Jothi, K.R., andVigneshwaran, A.2012. An Optimal Job Scheduling in Grid Using Cuckoo Algorithm. *International Journal of Computer Science and Telecommunications, 3(2), 65-69.*

Rabiee, M., and, Sajedi, H. 2013. Job Scheduling in Grid Computing with Cuckoo Optimization Algorithm. *International Journal of Computer Applications, 62(16), 38-44.*

Ritchie, G. 2003. Static multi-processor scheduling with ant colony optimization and local search. *Master thesis, School of Informatics, Univ. of Edinburgh.*

Ritchie, G., and Levine, J. 2003. A fast effective local search for scheduling independent jobs in heterogeneous computing environments. *Technical report, Centre for Intelligent Systems and their Applications, University of Edinburgh.*

Salman, A., Ahmad, I., and Al-Madani, S. 2002. Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems, 26(8), 363-371.*

Schopf, J.M. 2004. Ten actions when grid scheduling. *Grid Resource Management (Book), Springer, 64, Chapter 1.*

Talbi, E. 2002. A taxonomy of hybrid meta-heuristics. *Journal of Heuristics, 8(5), 541-564.*

Xhafa, F. 2007. A Hybrid Evolutionary Heuristic for Job Scheduling in Computational Grids. *Studies in Computational Intelligence, Springer, 75, Chapter 10.*

Xhafa, F., and, Abraham, A. 2010. Computational models and heuristic methods for Grid scheduling problems. *Future Generation Computer Systems, Elsevier, 26, 608-621.*

Xhafa, F., Alba, E., Dorronsoro, B., and Duran, B. 2008. Efficient batch job scheduling in grids using cellular memetic algorithms. *Journal of Mathematical Modelling and Algorithms, 7(2), 217-236.*

Xhafa, F., Duran, B., Abraham, A., and Dahal, K.P. 2008. Tuning struggle strategy in genetic algorithms for scheduling in computational grids. *Neural Network World, 18(3), 209-225.*

Yarkhan, A., and, Dongarra, J. 2002. Experiments with scheduling using simulated annealing in a Grid environment. In *Proceedings of GRID-2002, 232-242.*

Zhang, L., Chen, H., Sun, R., Jing, S., and Yang, B. 2008. A Task Scheduling Algorithm Based on PSO for Grid Computing. *International Journal of Computational Intelligence Research, 4(1), 37-43.*

Zomaya, A.Y., and, Teh, Y.H. 2001. Observations on using genetic algorithms for dynamic load-balancing. *IEEE Transactions on Parallel and Distributed Systems, 12(9), 899-911.*

Tarun Kumar Ghosh received his B.Tech. in Computer Science and Engineering from University of Calcutta, India; M.E. in Computer Science and Technology from B E College (DU) [now IIEST], Shibpur, Howrah, India and currently pursuing Ph.D. in Computing from University of Kalyani. He is currently an Associate Professor in the Department of Computer Science and Engineering, Haldia Institute of Technology, West Bengal, India. His research interests include Grid Computing, Optimization, Computer Architecture and Interconnection Networks. He has published in national and international journals, conference proceedings as well as books published by Tata McGraw-Hill.

Dr. Sanjoy Das received his both B. E. and M. E. in Mechanical Engineering and Ph. D. in Engineering from Jadavpur University, Kolkata, India. He has more than 18 years of teaching experience and is currently an Associate Professor in the Department of Engineering and Technological Studies, University of Kalyani, India. His research interests include Optimization techniques and Tribology.