BPISG: A Batching Heuristic Scheduling Algorithm With Taking Index Parameters for Mapping Independent Tasks on Heterogenous Computing Environment

Arash Ghorbannia Delavar , Ali Reza Khalili Boroujeni and Javad Bayrampoor

Department of Computer, Payame Noor Universtiy, PO BOX 19395-3697, Tehran, Iran

Abstract

In this paper we consider the problem of allocating independent heterogeneous tasks on grid environment in which A new batching heuristic scheduling algorithm with Taking index parameters will be presented. Reference to tasks compute scheduling, several methods are evaluated. With compare Min-mean and Improved Min-mean algorithms, we can create the specific situation that BPISG algorithm can be more efficient and more dependable than similar than previous algorithms. With taking the round time, consist acknowledgement and return time parameters, specific functionality has been created.

With implementation these parameters in the simulate environment, we can create situation that scheduling task will be done with better position and achieve high performance on computational grids. Finally the experiment and simulated results will show that the BPISG heuristic scheduling algorithm performs significantly to ensure high throughput, reduced makespan and more efficient in the grid environment.

Keywords: Grid Computing, Job Scheduling, Heuristic Algorithm, Load Balancing.

1. Introduction

Grid computing environment includes of different interconnected machines by interface networks to execute different tasks that have diverse computational requirements. In distributed systems such as grids, in which there are various sources, deciding about regularity of task and selecting the computing machines is an important problem in grid environments. The main purpose of grid systems is optimize using sources and maximizing the efficiency of the system. Managing various resources and task scheduling in grid environment are challenging and indispensable works [1].

Grid environments have tried to study various scheduled algorithms for reaching to above purposes. Scheduling is an important tool for this purpose. Tasks scheduling is an NP-complete problem and finding the absolute optimal solution is too hard. So many heuristics have been developed to solve this hard problem. The heuristic scheduling can be classified into two categories: on-line mode and batch-mode heuristics. In the on-line mode heuristics, a task is mapped on to a machine as soon as it arrives at the scheduler. In the batch-mode heuristics, tasks are not mapped on to machines as they arrive; instead, they are collected into the buffer and then it is scheduled at prescheduled time [3, 4].

In this paper, the study is based on the batch-mode heuristics and presents a new heuristic scheduling algorithm named BPISG to acquire efficient static mapping of meta-tasks to the machines in heterogeneous computing systems. The primary objects of the paper are the throughput maximization and reduced makespan (measure of the throughput) of the heterogeneous grid computing systems. With taking the round time index parameter in simulated environment, BPISG heuristic scheduling algorithm performs more efficiently in the grid environment. The proposed heuristic uses the benchmark model of Braun et al. [2, 3].

2. Related Works

Many heuristics algorithms have been designed and developed to solve meta-tasks optimal scheduling In distributed heterogeneous computing systems. Two sample of these heuristic algorithms are defined as follows:

2.1 Min-min

Min-min algorithm starts with a set of all unmapped tasks. The completion time for each task on each machine is calculated. The machine that has the minimum completion time for each job is selected. Then the job with the overall minimum completion time is selected and mapped to the machine. Again, this process is repeated with the remaining unmapped tasks [2, 3].

2.2 Min-mean

The heuristic scheduling algorithm Min-mean works in two phases.
• In phase 1, the task allocation is done based on the Min-min algorithm.
• In phase 2, the mean of all machines completion time is taken. The machine whose completion time is greater than the mean value is selected. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value [3].

2.3 Improved Min-mean Algorithm

Like to Min-mean Algorithm, Improved Min-mean Algorithm works also in two phases.

• In phase 1, the job allocation is done based on the Min-min algorithm.
• In phase 2, the mean of all machines completion time is taken. Then a small constant value Δt=7% is added to the mean completion time *(k = MeanCT + Δt).* This causes an improvement over Min-mean heuristic scheduling algorithm. After that the machine whose completion time is greater than the k value, is selected and The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the k value [4].

3. Problem Definition

The problem of task scheduling, will be studied in heterogeneous computing environment. The experimental study is based on a benchmark simulation model by Braun et al. [2].

In this model, there is number of independent tasks to allocation and number of machines to execute these tasks. Because of Nonpreemptive scheduling, each machine executes one task at a time. For static mapping, number of tasks, the size of the tasks and the number of machines in the heterogeneous computing environment are known a priori. Since there are static heuristics, the accurate estimate of the expected execution time for each task on each machine is known to execution and is contained within an ETC (expected time to compute) matrix where ETC(ti, mj) is an estimated execution time of task i on machine j. The size of ETC is t * m, where t, represents the number of the tasks and m, represents the number of the machines [3, 4].

ct_{ij} – Expected completion time of the task t_i on the machine m_j .

et_{ij} – Expected execution time of the task t_i on the machine m_j. Suppose that m_j has no load when task t_i is assigned.

r_j – Ready time of the machine m_j (the time when machine mj becomes ready to execute task t_i).

at_{ij} – acknowledgement time of the task t_i from the machine m_j (the wait time needed to mapping task t_i to the machine m_j).

rt_{ij} – Return time of the task t_i from the machine m_j (the wait time needed to return results of task t_i from machine m_j).

$$ct_{ij} = et_{ij} + r_j + at_{ij} + rt_{ij} \quad (1)$$

The primary object of BPISG heuristic scheduling algorithm is to minimize the makespan which is defined as completion time of the system:

$$\text{makespan} = \max(ct_{ij}). \quad (2)$$

3.1 BPISG Algorithm

The model of BPISG scheduling algorithm is based on the expected time to compute (ETC) matrix of Braun et al. [2]. The BPISG heuristic scheduling algorithm is defined as follows:

First:
do until all tasks t_i in meat-tasks M_t are scheduled
- for each task in M_t compute the earliest completion time and the machine that obtains it
- find the task t_k with the minimum earliest completion time
- assign each task t_k to the machine m_k giving the earliest completion time
- delete task t_k from M_t
- update r_j
- update ct_{ij} for all i

enddo

Second:
Sort all machine on minimum earliest completion time

Third:
for all machine m_j with the minimum earliest completion time
- for all task t_i selected with this machine m_j in phase 1
  - find the machine m_k with the minimum earliere completion time than mj
  - assign task t_i to the machine m_k and delete task t_i from m_i
  - reschedule task t_i on machine m_k
- endfor

endfor

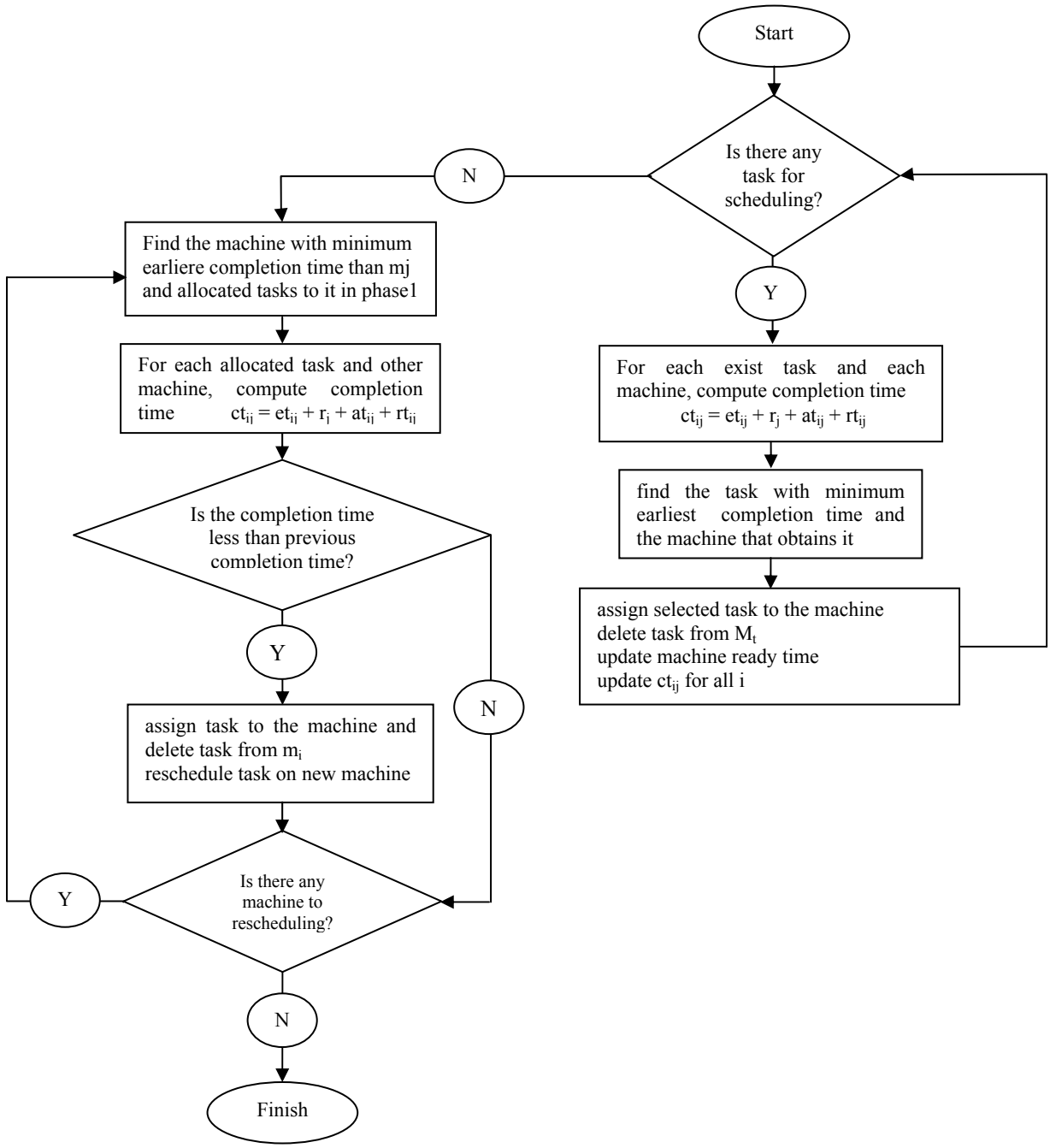

Figure 1. BPISG heuristic scheduling algorithm diagram

4. Benchmark Descriptions

To better evaluate the behavior of mapping heuristics, a model of the execution times of the tasks on the machines is needed; so that the parameters of this model can be changed to investigate the performance of the heuristics under different heterogeneous computing systems and under different types of tasks to be mapped. One such model consists of an expected time to compute (ETC) matrix, where contains the estimates for the expected execution times of a task on all machines, for all the tasks that are expected to arrive for service over a given interval of time.

The ETC model presented here can be characterized by three parameters: machine heterogeneity, task heterogeneity, and consistency. Four categories were proposed for the ETC matrix in: (a) high task heterogeneity and high machine heterogeneity, (b) high task heterogeneity and low machine heterogeneity, (c) low task heterogeneity and high machine heterogeneity, and (d) low task heterogeneity and low machine heterogeneity.

The ETC matrix can be further classified into three categories, consistent, inconsistent and partially-consistent. For a consistent ETC matrix, if a machine m_x has a lower execution time than a machine m_y for a task t_k, then the same is true for any task t_i.

In inconsistent ETC matrices, the relationships among the task computational requirements and machine capabilities are such that no structure as that in the consistent case is enforced.

A combination of these two cases, which may be more realistic in many environments, is the partially-consistent ETC matrix, which is an inconsistent matrix with a consistent sub-matrix [5].

The experimental results are classified into 12 different types of ETC matrices includes three groups of four instances each. Every instance consists of 64 tasks and 16 machines. The first group relates to the inconsistent ETC matrices. The second and third group relates to the consistent and partially-consistent ETC matrices. All instances based on the three metrics: task heterogeneity, machine heterogeneity and consistency.

the round time, consist acknowledgement and return time parameters for each of the instances with a discrete uniform random function which are created as the 32 byte packets size. the acknowledgement time of each packets is between 1 to 200 ms and the return time of the same packet is vary between 0 to 0.4 of acknowledgement time of that packet.

5. Performance Analysis

To evaluate the efficiency of propose algorithm , the BPISG heuristic scheduling algorithm is compared with Min-mean and Improved Min-mean heuristic algorithm in all the three group of four instances.

The follow tlables and diagrams show the improvement in percentage of the BPISG heuristic scheduling algorithm over Min-mean and Improved Min-mean heuristic scheduling algorithm.

Table 1: Improvement of BPISG algorithm over Min-mean and Improved Min-mean

Consistency	Improved Over Min-mean		Improved Over Improved Min-mean	
Inconsistent	High-High	15.24 %	High-High	15.24 %
	High-Low	3.99 %	High-Low	0.95 %
	Low-High	19.13 %	Low-High	9.14 %
	Low-Low	19.54 %	Low-Low	19.54 %
Consistent	High-High	1.27 %	High-High	1.27 %
	High-Low	11.37 %	High-Low	2.92 %
	Low-High	0.00 %	Low-High	0.00 %
	Low-Low	9.93 %	Low-Low	9.93 %
Partially Consistent	High-High	20.87 %	High-High	20.87 %
	High-Low	14.49 %	High-Low	14.49 %
	Low-High	21.26 %	Low-High	21.26 %
	Low-Low	8.33 %	Low-Low	8.33 %

Table 2: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,high machine in Inconsistent type

Instance in Inconsistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-High	5458450	5458450	4736396

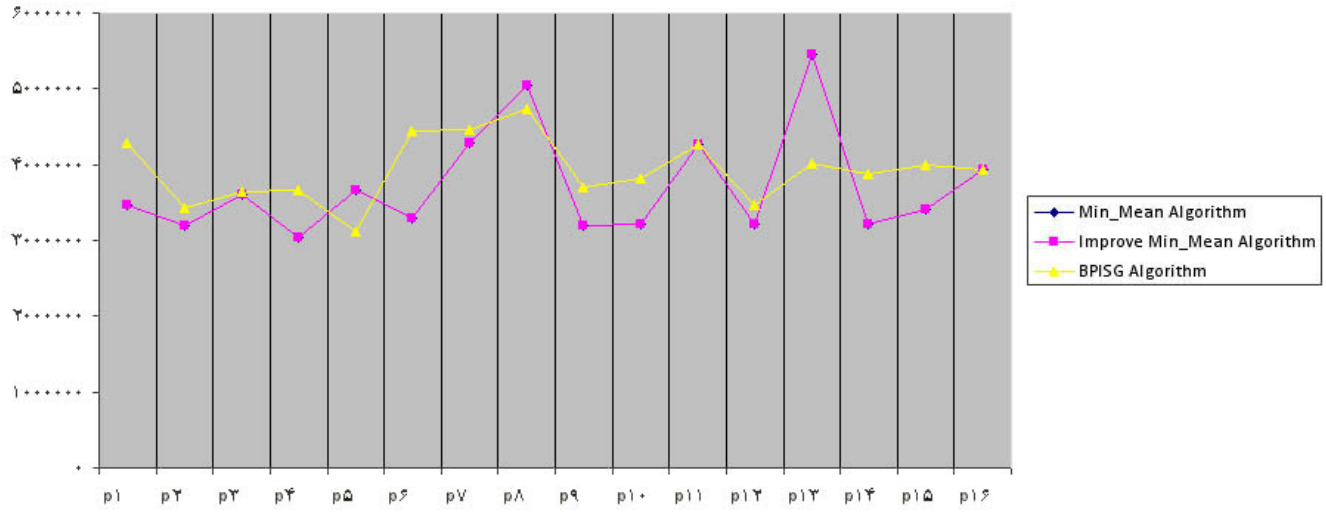

Figure 2. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & high machine in Inconsistent type with 15.24% improvement more than Min-mean and Improved Min-mean

Table 3: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,low machine in Inconsistent type

Instance in Inconsistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-Low	1930096	1873779	1856117

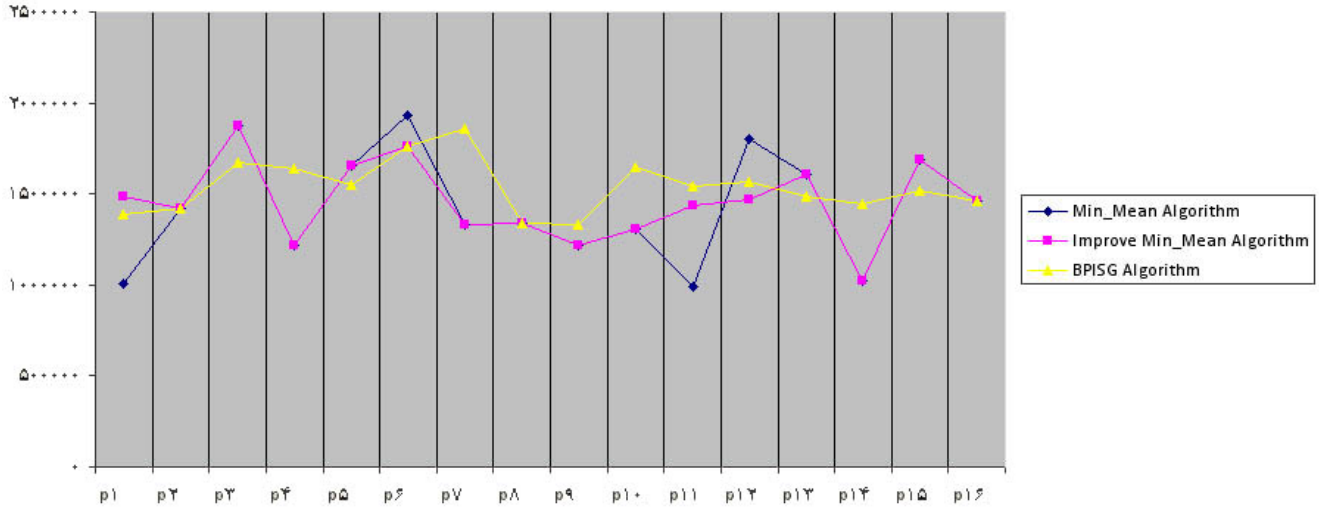

Figure 3. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & low machine in Inconsistent type with 3.99 % improvement more than Min-mean and 0.95 % in compare with Improved Min-mean

Table 4: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,high machine in Inconsistent type

Instance in Inconsistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-High	1084290	993399	910166

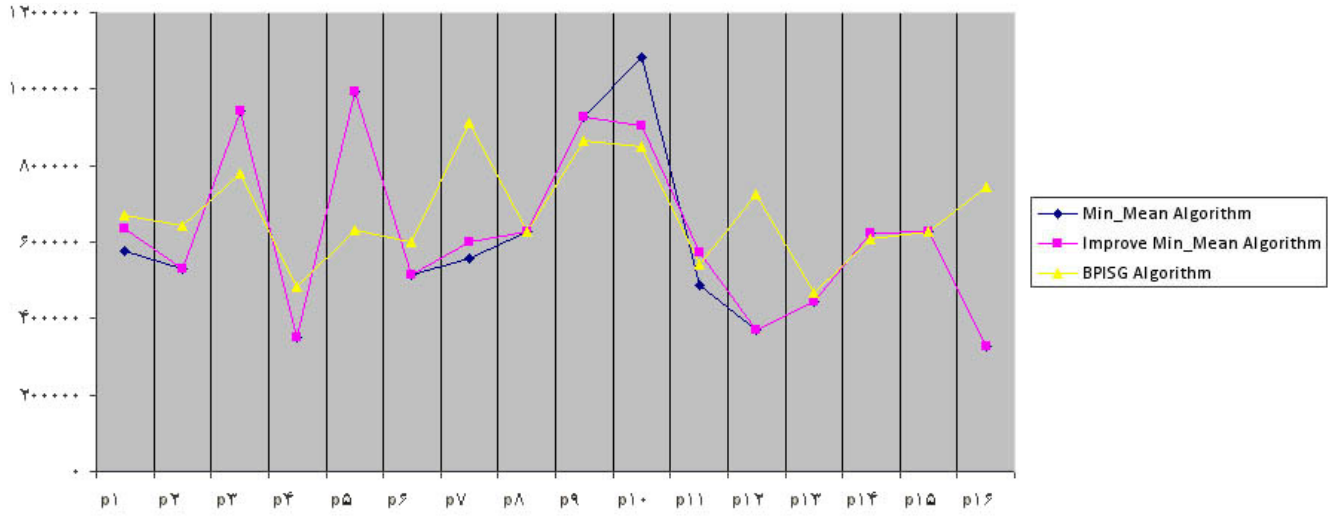

Figure 4. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low task & high machine in Inconsistent type with 19.13 % improvement more than Min-mean and 9.14 % in compare with Improved Min-mean

Table 5: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,low machine in Inconsistent type

Instance in Inconsistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-Low	1389099	1389099	1162040

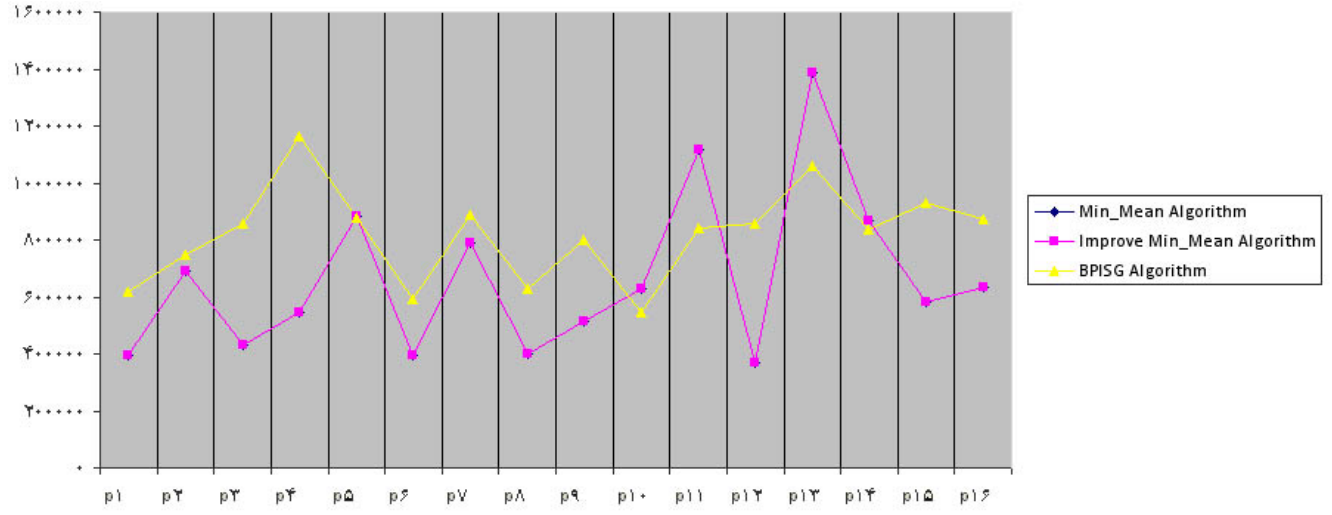

Figure 5. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low & task low machine in Inconsistent type with 19.54 % improvement more than Min-mean and Improved Min-mean

Table 6: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,high machine in Consistent type

Instance in Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-High	10644873	10644873	10511821

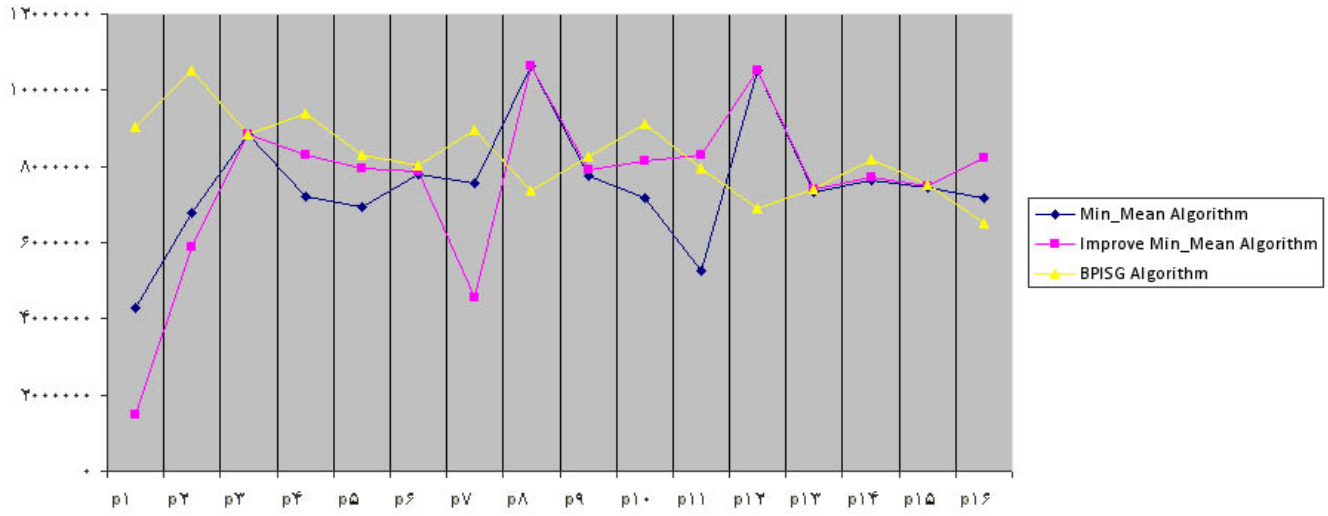

Figure 6. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & high machine in Consistent type with 1.27 % improvement more than Min-mean and Improved Min-mean

Table 7: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,low machine in Consistent type

Instance in Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-Low	3723622	3441148	3343366

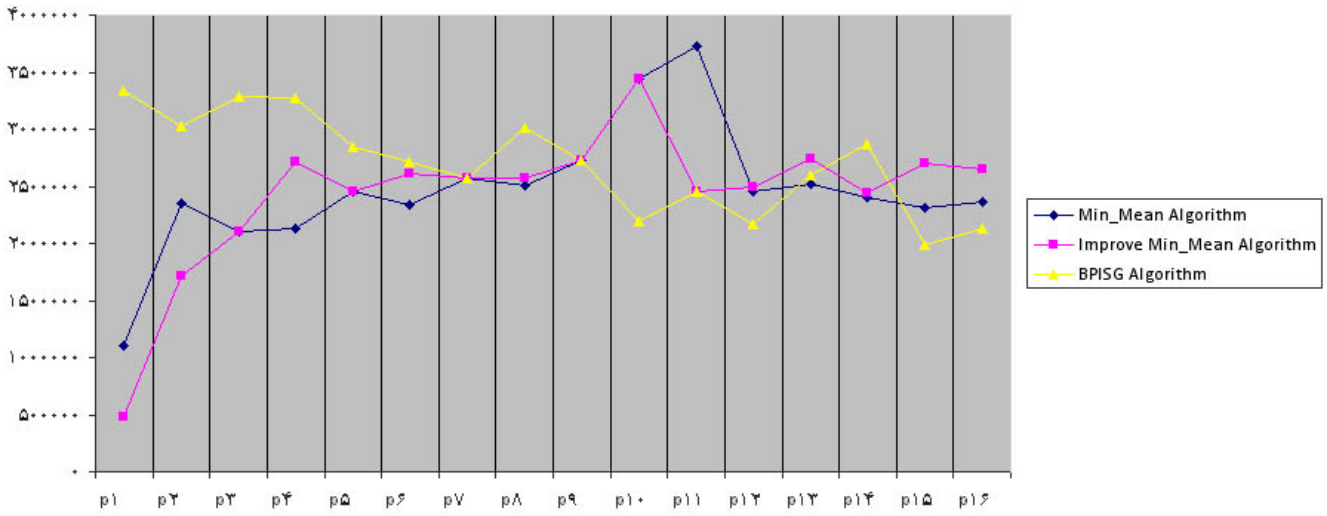

Figure 7. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & low machine in Consistent type with 11.37 % improvement more than Min-mean and 2.92 % in compare with Improved Min-mean

Table 8: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,high machine in Consistent type

Instance in Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-High	1799023	1799023	1799023

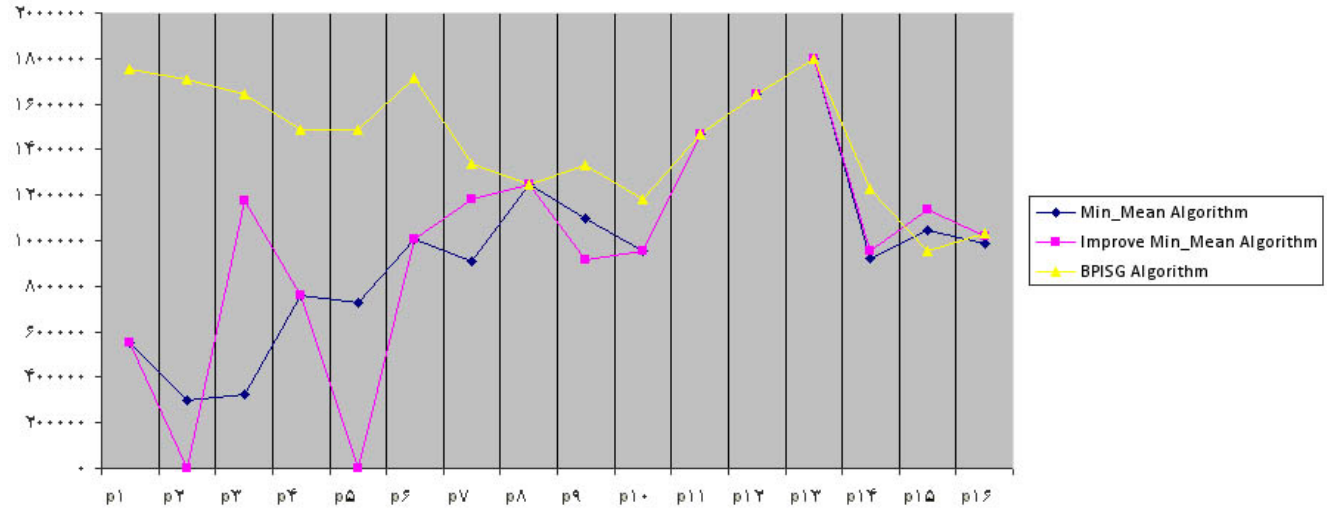

Figure 8. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low task & high machine in Consistent type with 0.00 % improvement more than Min-mean and Improved Min-mean

Table 9: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,low machine in Consistent type

Instance in Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-Low	1945199	1945199	1769463

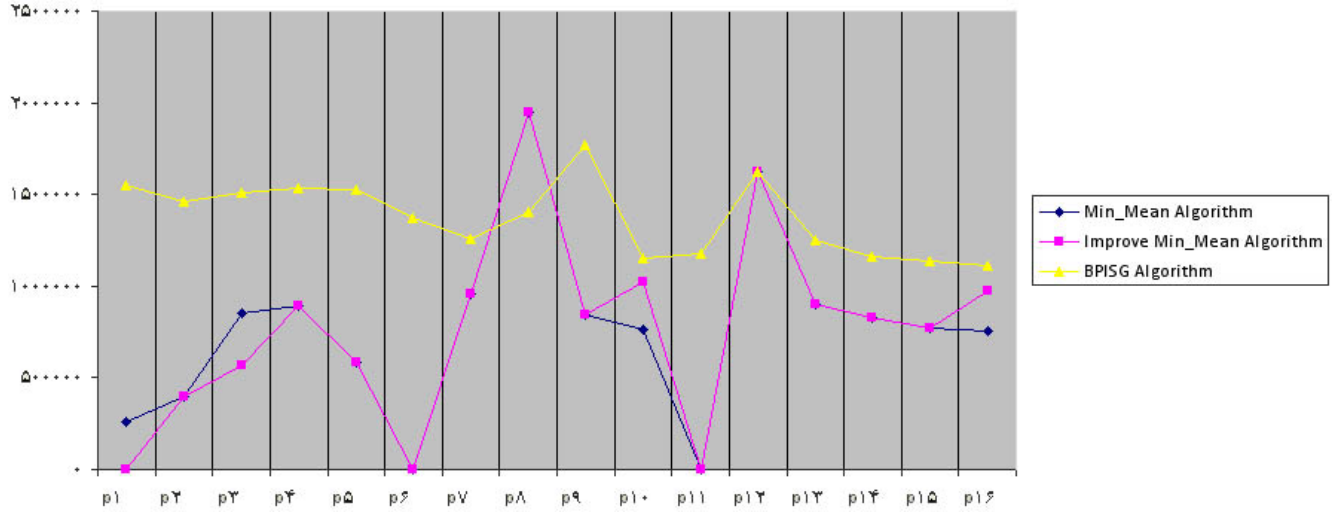

Figure 9. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low task & low machine in Consistent type with 9.93 % improvement more than Min-mean and Improved Min-mean

Table 10: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,high machine in partially-Consistent type

Instance in Partially-Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-High	7198299	7198299	5955583

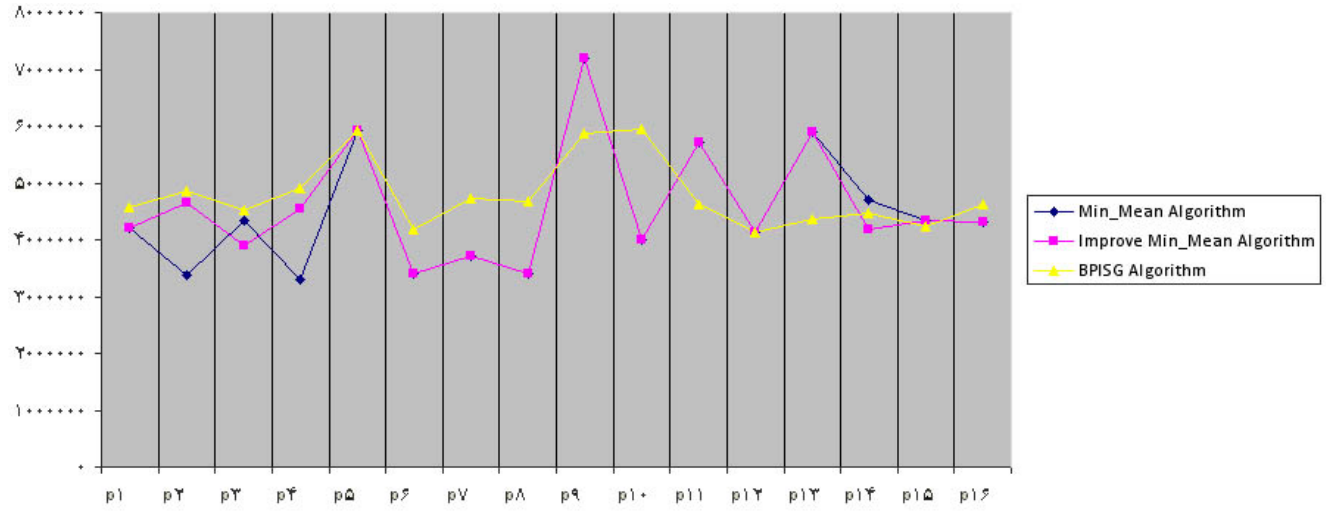

Figure 10. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & high machine in partially-Consistent type with 9.93 % improvement more than Min-mean and Improved Min-mean

Table 11: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for high task,low machine in partially-Consistent type

Instance in Partially-Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
High-Low	1967366	1967366	1718439

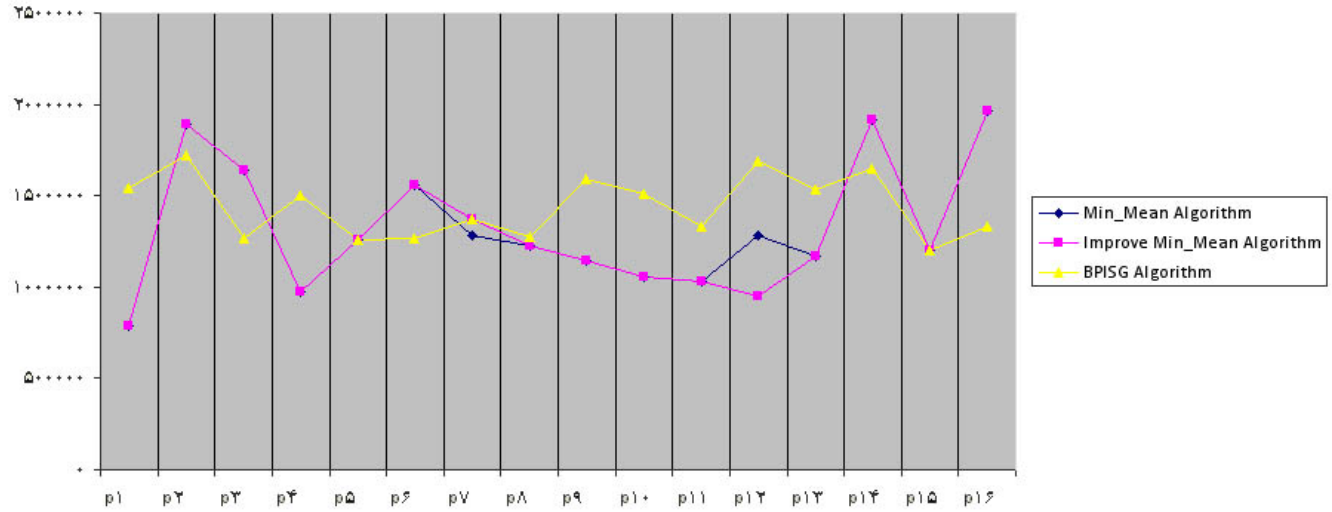

Figure 11. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for high task & low machine in partially-Consistent type with 14.49 % improvement more than Min-mean and Improved Min-mean

Table 12: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,high machine in partially-Consistent type

Instance in Partially-Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-High	828352	828352	683121

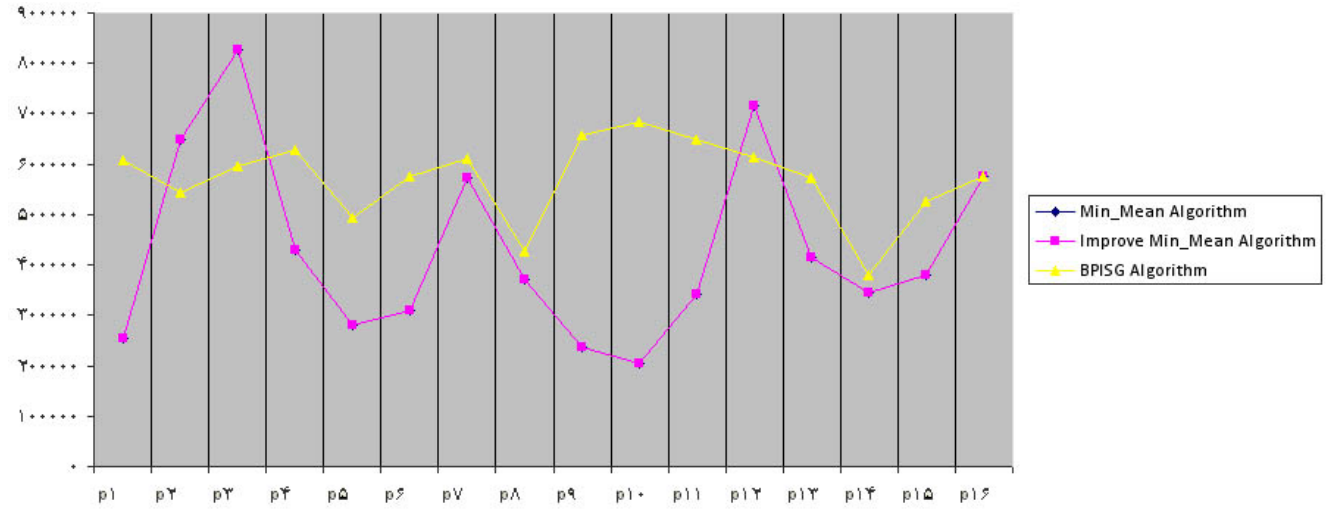

Figure 12. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low task & high machine in partially-Consistent type with 21.26 % improvement more than Min-mean and Improved Min-mean

Table 13: Comparison of makespan value obtained by min-Mean, Improved Min-mean and BPISG algorithm for low task,low machine in partially-Consistent type

Instance in Partially-Consistent	Min-mean	Improved Min-mean	BPISG Algorithm
Low-Low	1339542	1339542	1236585

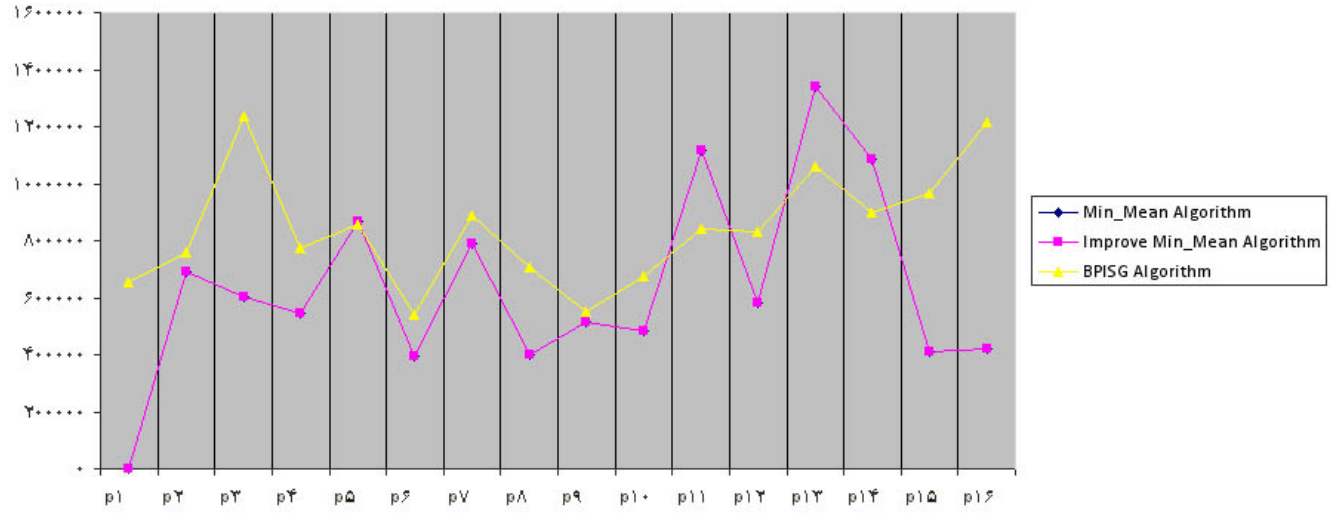

Figure 13. Comparison of makespan value obtained by Min-mean, Improved Min-mean and BPISG algorithm for low task & low machine in partially-Consistent type with 8.33 % improvement more than Min-mean and Improved Min-mean

6. Conclusions And Future Work

experimental results show the BPISG algorithm has average 12.12 and 10.33 % improvement more than Min-mean and Improved Min-mean algorithms. BPISG also reduced makespan, increase efficiency and achieve better mapping than previous algorithms in the grid environment. The future research will be focused on other index parameters, such as bandwdith and delay times in grid environment.

References

[1] A.Ghorbannia Delavar, M.Nejadkheirallah and M.Motalleb, "A New Scheduling Algorithm for Dynamic Task and Fault Tolerant in Heterogeneous Grid Systems Using Genetic Algorithm" , IEEE 2010.

[2] Tracy D.Braun, Howard Jay Siegel and Noah Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing 61, 2001, pp.810-837.

[3] Kamalam.G.K and Murali bhaskaran.V, "A New Heuristic Approach:Min-Mean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems", Journal of Computer Science and Network Security, January 2010.

[4] G. K. Kamalam and V. Murali Bhaskaran, "An Improved Min-Mean Heuristic Scheduling Algorithm for Mapping Independent Tasks on Heterogenous Computing Environment", Journal of Computational cognition, december 2010.

[5] Shoukat Ali, Howard Jay Siegel and Muthucumaru Maheswaran, "Task Execution Time Modeling for Heterogeneous Computing Systems", IEEE *Computer*, 2000.

Arash Ghorbannia Delavar received the MSc and Ph.D. degrees in computer engineering from Sciences and Research University, Tehran, IRAN, in 2002 and 2007. He obtained the top student award in Ph.D. course. He is currently an assistant professor in the Department of Computer Science, Payam Noor University, Tehran, IRAN. He is also the Director of Virtual University and Multimedia Training Department of Payam Noor University in IRAN. Dr.Arash Ghorbannia Delavar is currently editor of many computer science journals in IRAN. His research interests are in the areas of computer networks, microprocessors, data mining, Information Technology, and E-Learning.

Ali Reza Khalili Boroujeni received the BS, in 2000 and now, he is a Student the MS degree in the department of Computer Engineering and Information Technology in Payam Noor University, Tehran, IRAN. His research interests include computer networks, grid and scheduling algorithm.

Javad Bayrampoor received the BS, in 2007 and now, he is a Student the MS degree in the department of Computer Engineering and Information Technology in Payam Noor University, Tehran, IRAN. His research interests include computer networks, grid and scheduling algorithm.