A Balanced Scheduling Algorithm with Fault Tolerance and Task Migration based on Primary Static Mapping (PSM) in Grid

Arash Ghorbannia Delavar
Department of Computer,
Payame Noor University,
PO BOX 19395-3697,
Tehran, Iran

Ali Reza Khalili Boroujeni
Department of Computer,
Payame Noor University,
Tehran, Iran

Javad Bayrampoor
Department of Computer,
Payame Noor University,
Tehran, Iran

ABSTRACT
In this paper we present a balanced scheduling algorithm with considering the fault tolerance and task migration of allocating independent tasks in grid systems. Resource scheduling and its management are great challenges in heterogeneous environment. Hence load balancing is one of the best solutions to achieve the above purposes. The scheduling algorithm which we will present in follow, with taking the fault tolerance, checkpointing method, task migration and priority for mapping independent tasks on heterogeneous computing environment, creates the specific situation to ensure high performance in grid systems. So by implementing these parameters we can achieve more efficient and dependable performance than similar previous algorithms. It will be done with better condition and achieve high performance in computational grids in compare with Min-min algorithm. Finally the experiment and simulated results show that proposed balanced scheduling algorithm performs significantly to ensure high throughput, reduced makespan, reliability and more efficiency in the grid environment.

Keywords
Grid Computing, Task Scheduling, Heuristic Algorithm, Load Balancing, Fault Tolerance, Task Migration, PSM

1. INTRODUCTION
Grid is emerging as a wide scale infrastructure and next generation parallel and distributed computing to aggregates dispersed heterogeneous resources, support source sharing, providing services to fit needs of scientific applications, business, engineering and Commerce [1].
Grid computing environment combination of widely spread computational machines includes of different interconnected machines by interface network to execute different tasks that have diverse computational requirements. A grid involves a variety of resources that are heterogeneous naturally and might span several administrative domains across not narrow geographical distances. Grid computing environment includes of different interconnected machines by interface networks to execute different tasks that have diverse computational requirements. The main purpose of grid systems is optimize using sources and maximizes the efficiency of the system. Managing various resources and task scheduling in grid environment are challenging and indispensable works [2, 3].
Tasks scheduling is a NP- complete problem and finding the absolute optimum solution is too hard. So many heuristic algorithms 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 [4, 5].
Our study is based on the batch-mode heuristics, and presents a batching heuristic scheduling algorithm with consider the fault tolerance and task migration of dedicating independent tasks in grid systems. The scheduling algorithm which we will present in follow executes primary static mapping (PSM) of meta-tasks on the machines in grid systems. Then based on PSM the tasks will be mapped on the machines. The main idea is that if a fault occurs at run time ,or we need to migrate the tasks, the execution will be continued with switching from a processing node to another node, based on PSM (as an optimal target). In this proposed algorithm, the failed machines can be returned to systems to reallocating. By implement the fault tolerance and priority for mapping the tasks in simulated environment we will achieve more efficiently in proposed scheduling algorithm performance, throughput maximization and reduced makespan (measure of the throughput) of the heterogeneous grid computing systems in the grid environment.

2. Related Works
Many heuristics algorithms have been designed and developed to solve meta-task optimal scheduling in distributed heterogeneous computing systems. Braun et al. have studied the relative performance of eleven heuristic algorithms for task scheduling in grid computing. They have also provided a simulation basis for researchers to test the algorithms. The simple algorithms proposed by Braun are Opportunistic Load Balancing (OLB), Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-min, Max-min, Duplex, Genetic Algorithm (GA), Simulated Annealing (SA), Genetic Simulation Annealing (GSA), Tabu and A^* [6].
The Min-min heuristic begins with the set of all unmapped tasks. Then, the set of minimum completion times is found. Next, the task with the overall minimum completion time is selected and assigned to the corresponding machine. Last, the newly mapped task is removed from unmapped tasks set and the process repeats until all tasks are mapped. Min-min is based on the minimum completion time and considers all unmapped tasks during each mapping decision at a time. Their

results show that Min-min is the simple and fastest algorithm and its good performance depend on the choice of mapping the meta-tasks to the first choice of minimum execution time. However the drawback of Min-min is that, it is unable to balance the load because it usually assigns the small task first and few larger tasks, while at the same time, several machines sit idle, which leads to poor utilization of resources [6].

2.1 Load Balancing

The resource managers or usages must modify their behavior dynamically so as to extract the maximum performance from the available resources and services. So to achieve high performance we need to understand the factors that can affect the performance of an application like load balancing which is one of most important factors which influence the overall performance of application. Load balancing is a technique to enhance resources, using parallelism, exploiting throughput improvisation and to cut response time through an appropriate distribution of the usages [7, 8].

2.2 Fault Tolerance

Fault tolerance is an important problem in grid computing as the dependability factor of grid resources and may become more prevalent in grid applications. The appearance of grid computing further increases the importance of fault tolerance. Some of the factors due to which the probability of problem in a grid environment is much higher than a traditional distributed system are lack of centralized environment, predominant execution of long tasks, highly dynamic resource availability and diverse geographical distribution of resources and different nature of grid resources. Thus, fault tolerance related features must be used in grid task planning to improve the performance of the grid system [9].

2.3 Checkpointing Tools

When a failure occurs the whole application is shutdown and has to be restarted from the beginning. A technique to avoid restarting of the application from the beginning is rollback recovery which is based on the idea of checkpoint. It periodically saves the application's state to stable storage. So whenever a failure interrupts a volunteer computation, the application can be resumed from the last stable checkpoint. The tools of checkpointing can be classified into two types, kernel-level and user-level. Kernel-level checkpointing tools are a part of the operating system kernel, while user-level checkpointing tools are themselves application programs. Kernel-level checkpointing is often implemented through inter process communication mechanisms such as signals, making user-level checkpointing portable [9].

2.4 Task Migration

The definition of migration is the movement of process, job, data, method, or service from one node to another. We can aim process migration as a fail-safe mechanism. It is supposed to prevent running jobs or processes from being failed because of shutdown of the execution node, power failure of the area or personal factor in the management of the execution node. Main situations show that migration mechanism is on demand. However, to migration running job or process to other execution nodes may pay for something else, such as the late of the complete time of these jobs, because of communication time, checkpointing time and reschedule time. A simple equation can be used to estimate the total execution time of a job which accounts from when a user submits the job to when the job is completed or failed [10,11]. A general migration mechanism includes:

- **Shared Storage devices:**

Shared storage devices are the most common methods used in a migration mechanism. Once the system discover load unbalance of some processors over specific threshold, it migrate the tasks or jobs in the waiting queue and dispatches to the most suitable idle processor through shared-storage devices, which act as a media for the storage of process states or images [11].

- **Preserving Memory image:**

Preserving memory image is the activity of writing the states of a running process to a file. Checkpointing is a general term referred to collecting and keeping the states of a running process, which is an operation of capturing the states and data of a running job or process. The items of captured states and data of a running process include:

-Registers containing the address, variables, and data else.

-Memory spaces keeping source codes, libraries, data structures.

-Files containing data with a large size [11].

In the next section, we will describe the details of the algorithm and show the benefits of our work via comparison and simulation. Finally, we will conclude our contributions and point out the future work scheduling algorithm.

3. Problem Definition

The problem of task scheduling will be studying in heterogeneous computing environment. In this environment, there is number of independent tasks to allocate and number of machines to execute these tasks and each machine executes one task at a time. For this mapping, a number of tasks, machines, the machine instructions for each task, the processing speed of machines, the transmission size and the return result size of task file and the network bandwidth between the scheduler and the machines are known and there for the accurate estimate of the expected execution time for each task on each machine will be known to execution. The fault tolerance and task migration mechanism that is used in proposed algorithm, rescheduling tasks which have failed or delayed with switching from a processing node to another node based on PSM.

C_{ij}^{t} – Expected completion time of the task t_i on the machine m_j.

E_{ij}^{t} – Expected execution time of the task t_i on the machine m_j. (Machine instructions of the task i / processing speed of the machine j) Suppose that machine m_j has no load when task t_i is assigned.

R_j^t : Ready time of the machine m_j (the time when machine m_j becomes ready to execute task t_i).

F_i^t : The transmission size of the task file i.

F_i^r : The return result size of task file i.

B_j^w : The network bandwidth between the scheduler and the machine j.

T_{ij}^{t} : Transmission time of the task file t_i to the machine m_j (the wait time needed to mapping task t_i to the machine m_j).

R_{ij}^{t} : Return time of the task file t_i from the machine m_j (the wait time needed to return results of task t_i from machine m_j).

$$C_{ij}^{t} = E_{ij}^{t} + R_j^t + T_{ij}^{t\&r_{ij}} \quad (1)$$

In Which:

$$T_{ij}^{t\&r_{ij}} = (F_i^t + F_i^r)/B_j^w \quad (2)$$

Or:

$$T_{ij}^{t\&r_{ij}} = T_{ij}^{t} + R_{ij}^{t} \quad (3)$$

Ensure high throughput when a fail occurs and reduced makespan, is primary object of proposed heuristic scheduling

algorithm in this paper. Makespan is defined as completion time of the system:

$$Makespan = Max(C_{ij}^{t}) \quad (4)$$

3.1 Proposed Algorithm

The proposed scheduling algorithm execute at two phases. First; presents a static mapping of meta-tasks on the machines in heterogeneous computing systems based on Min-min. Second; for more workload balancing and decrease system makespan, tasks are rescheduling on machines again. The above phases are defined as primary static mapping (PSM). Then based on PSM the tasks start to mapping on the machines. If a fail occurs at run time, by using checkpointing method, tasks which have been failed can be continue with switching from a processing node to another node that has minimum completion time based on PSM.

In our proposed algorithm we have these restricts:

1. The proposed strategy is based on message transmission.
2. The dynamic load balance is used at user-level.

The proposed heuristic scheduling algorithm is defined as follow:

First:

Do until all tasks in meat-tasks are scheduled

- For each task
  - For each machine
    - Compute the earliest completion time
    - Find the task with the minimum earliest completion time
    - Assign each task to the machine giving the earliest completion time
    - Delete task from meat-tasks
    - Update machine ready time
  - End for
- End for

End do

Second:

For all machine order by minimum earliest completion time respectively

- For all tasks have selected with this machine in first phase
  - Find the another machine with the minimum earliest completion time than previous machine
  - Reassign task to the new machine
  - Delete task from previous machine
  - Update machines ready time
- End for

End for

The proposed heuristic algorithm phases diagram as PSM is showed as follow in Figure 1.

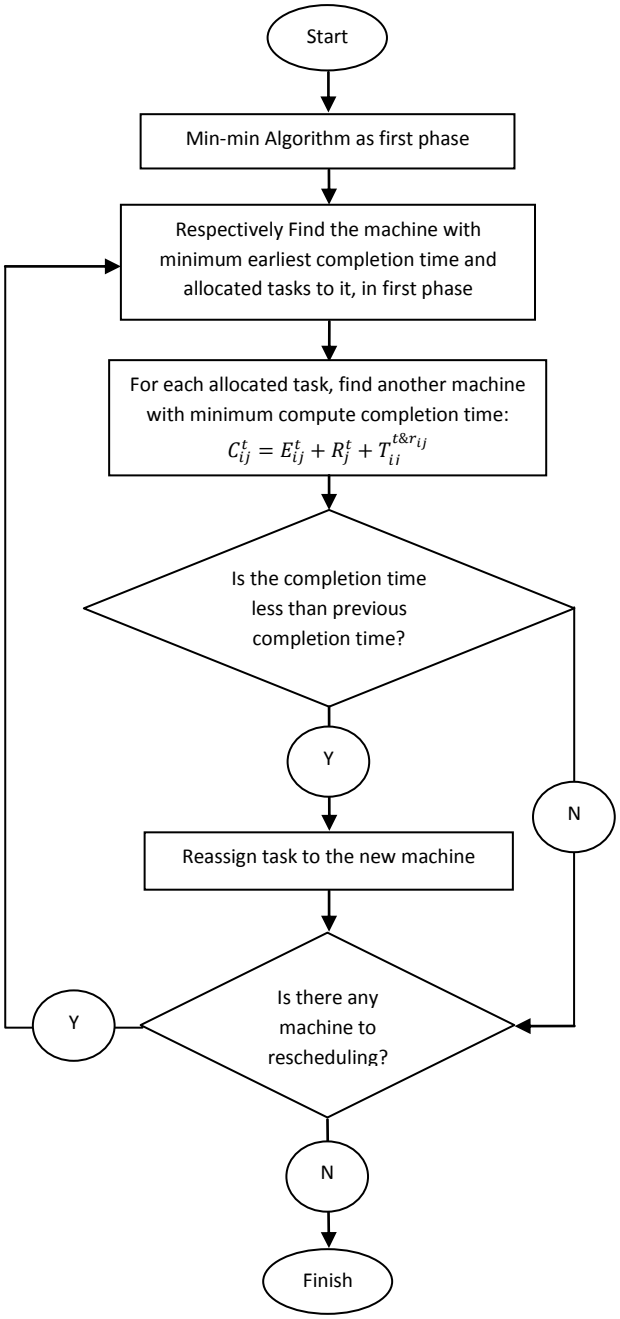

Fig1. The proposed algorithm diagram

Algorithm starts based on PSM as static scheduling and will be continue while no fail occurs. So if no fail occurs, at the end based on PSM, Tasks will become mapped on corresponding machine. At follow we will describe that if a fail occurs, the tasks scheduling will change to dynamic scheduling to reassign failed tasks to a new machine based on PSM, and checkpointing method used for failed running tasks.

3.2 Fault Tolerance Implementation

When a fail occurs, if the task is running, continuance of calculation can be save to keep on its task with switching from a processing machine that is failed to another machine not failed and has minimum completion time base on PSM. Then, completion times of machines updated and saved to probable failed states in future. This processing will continue while schedule/reschedule whole tasks on machines will be done. Each machine executes a single task at a time and in an order in which the tasks are assigned and in failed condition, failed task will reassign to selected machine after its running task. By using checkpointing mechanism, the return time will be improved considerably that is useful for the environments with high source fault rate.

In the experiments, omission faults will arise when resources become unavailable, due to the dynamic nature of many grids. The fault tolerance proceeding as rescheduling processing for failed tasks is defined as follow:

- Determine the failed machine and the task that must be reallocated on a new machine based on PSM
- If the failed task is running
  - Save continuance of calculation

  End if
- Switching the task to another machine which has minimum completion time ($C_{ij}^t = E_{ij}^t + R_j^t + T_{ij}^{t\&r_{ij}}$) base on PSM after the current running task
- Update primary static mapping information (PSMI) to probable failed states in future

3.3 Checkpointing Tool Implementation

Checkpointing is combination of two activities that save the running data and restore it after getting suitable resource. Captured states and data of a running process include registers containing the address, variables, memory spaces keeping source codes, libraries, data structures, files containing data with a large size and other data.

In the experiments, we adopt a user-level checkpointing tool, which is a kind of usage consist a set of libraries and programs for checkpointing.

3.4 Task Migration Implementation

In this paper we assume the information collected by the grid monitor system that is based on periodical framework (where a period is called a time slice). In each time slice, the system will record the availability of all nodes. So the task scheduling system is based on the statistical information gathered by the system monitoring and the estimated migration cost. Similar to a checkpoint/restart system, the migration is separated into three phases: data collection, data transmission and data restoration. The times spent on these phases are represented as T_c, T_t, and T_r respectively. The source machine and the destination machine are represented as m_s and m_d. For a general process migration system without any optimization, the cost to migrate a running process from m_s to m_d is:

$$C_{sd} = T_c + T_t + T_r \qquad (5)$$

Given an application App running on a machine m_s, at time $t = 0$, it reaches a poll point P. If App does not migrate at time t, it finishes on m_s at t_s. If App is scheduled to migrate to another machine m_d at time t, it finishes on m_d at t_d. The available communication bandwidth from m_s to m_d is B_{sd} which can be estimated with existing network performance prediction tools. The available computing capacity of m_s and m_d are represented as P_s and P_d sequence.

The migration cost C_{ij} is defined as the time spent to migration App from m_s to m_d:

$$t_d = C_{sd} + t_s * P_s/P_d \qquad (6)$$

So if m_d has the same computing capacity as m_s, that is $P_d = P_s$ (in most cases, this means m_d is identical to m_s), then the migration cost is:

$$C_{ij} = t_d - t_s \qquad (7)$$

The proposed algorithm strive to move jobs when wait time at the machines rise above specific threshold. So if the wait time of the task is below threshold τ, the system volunteers itself for receiving jobs by informing other machines of its low utilization. The migration threshold τ also acts as a gate to discourage excessive job movement. We define threshold τ as:

$$\tau = Max\left(C_{ij}^t\right) + \Delta t \qquad (8)$$

In Which:

$Max\left(C_{ij}^t\right)$ – Maximum completion time of the task t_i on the machine m_j based on PSM.

Δt – Variable value is defined as follows:

$$\Delta t = \sum R_m \,/\, \sum R_{m_i} \qquad (9)$$

(We define Δt as a variable parameter in order to preventing from over migration of tasks and allocating greater chance to machines with Maximum completion time, based on PSM.)

Where:

$\sum R_m$ – Sum of task computational requirements that is assigned to whole machines.

$\sum R_{m_i}$ – Sum of task computational requirements that is assigned to machine m_j.

In the experiments, the delay of the complete time of these jobs occurs by communication time because of bandwidth between the scheduler and the machines and execution time as a result of Machine capabilities. The task migration proceeding as rescheduling processing for tasks with long delay is based on threshold that is defined as follow:

- Determine the machine which cannot be executed on deadline based on specific threshold and the task that must be reallocated on a new machine base on PSM
- Restore the last checkpointing calculation of the running task
- Switching the task to another machine which has minimum completion time ($C_{ij}^t = E_{ij}^t + R_j^t + T_{ij}^{t\&r_{ij}}$) base on PSM after the current running task
- Update primary static mapping information (PSMI) to probable failed states in future

4. Benchmark Descriptions

To better evaluate the behavior of mapping heuristics, investigating the performance of the heuristics under different heterogeneous computing systems and under different types of tasks must be mapped. So the expected execution time for each task on each machine can be achieved from machine instructions of the task, the processing speed of the machines, the transmission size of the task file, the return result size of task file and the network bandwidth between the scheduler/machine in grid.

5. Performance Analysis

To evaluate the efficiency of the proposed algorithm, it is compared with Min-min heuristic algorithm in fault states and delay times with checkpointing and without checkpointing method. The follow tables show the parameters of system, machines and tasks. Also follow Diagrams show the improvement of proposed heuristic scheduling algorithm over Min-min at different percentage of failure coefficient.

Table 1. The used parameters for simulating of proposed algorithm with fault tolerance and Min-min algorithm

Number of tasks	512
Number of machines	16
Task computational requirements	5 - 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	0, 10, 20, 30 and 40 (%)
Delay Rate	0

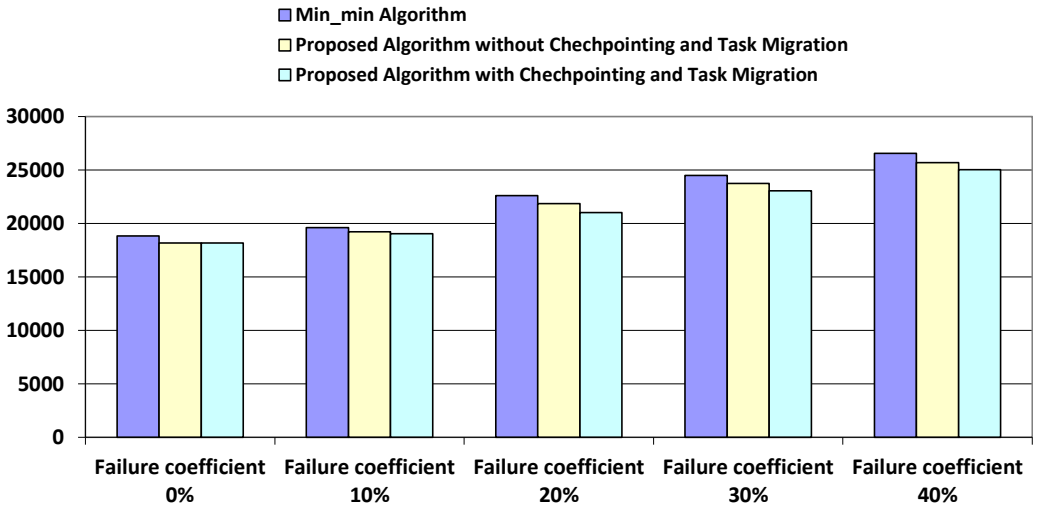

Fig2. Comparison of result obtained by Min-min algorithm, proposed algorithm without checkpointing and proposed algorithm with checkpointing for various fault occurrence rates based on table 1

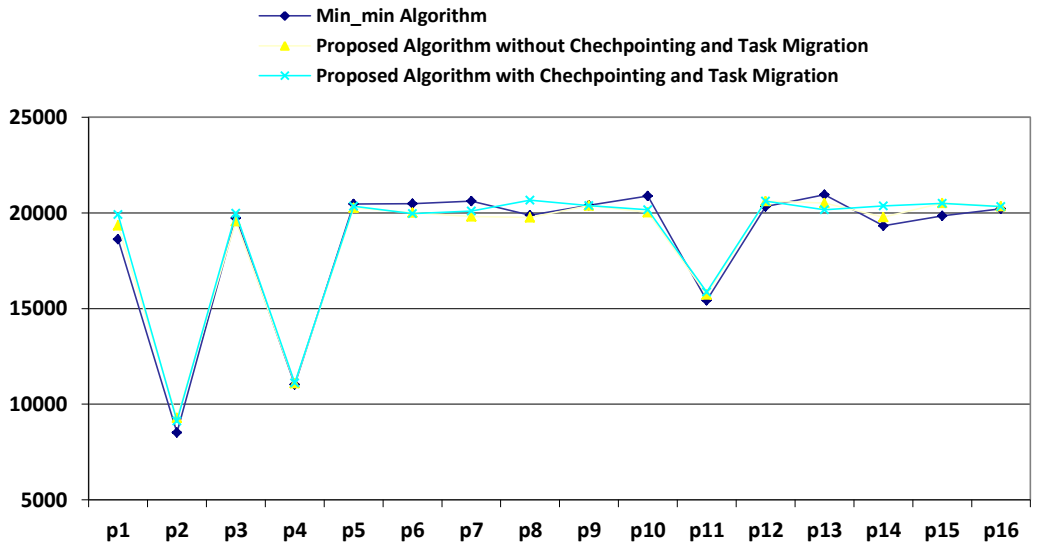

Fig3. Comparison of result obtained by Min-min algorithm, proposed algorithm without checkpointing and proposed algorithm with checkpointing for 20% fault occurrence rate based on table 1

Table 2. The used parameters for simulating of proposed algorithm with fault tolerance and Min-min algorithm

Number of tasks	1024
Number of machines	16
Task computational requirements	5 – 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	0, 10, 20, 30 and 40 (%)
Delay Rate	0

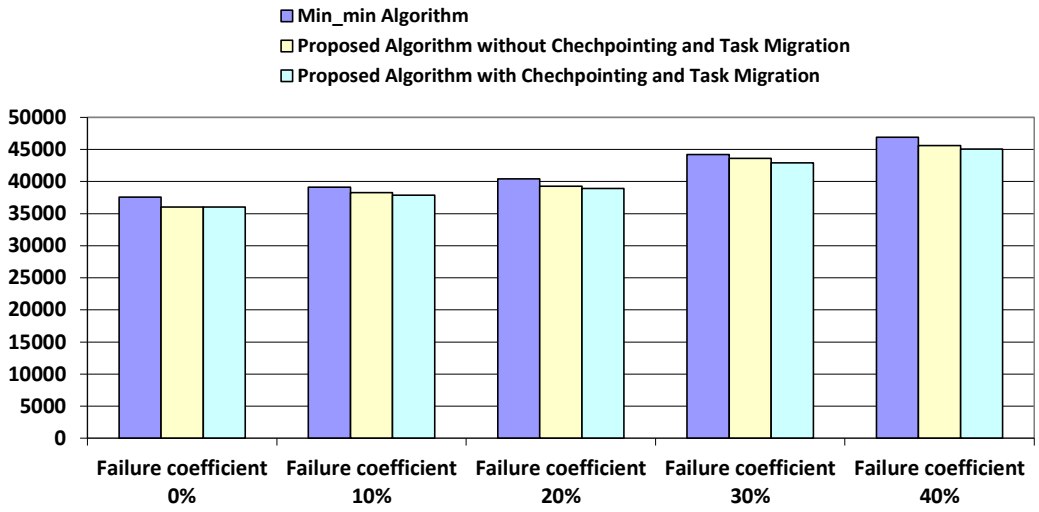

Fig4. Comparison of result obtained by Min-min algorithm, proposed algorithm without checkpointing and proposed algorithm with checkpointing for various fault occurrence rates based on table 2

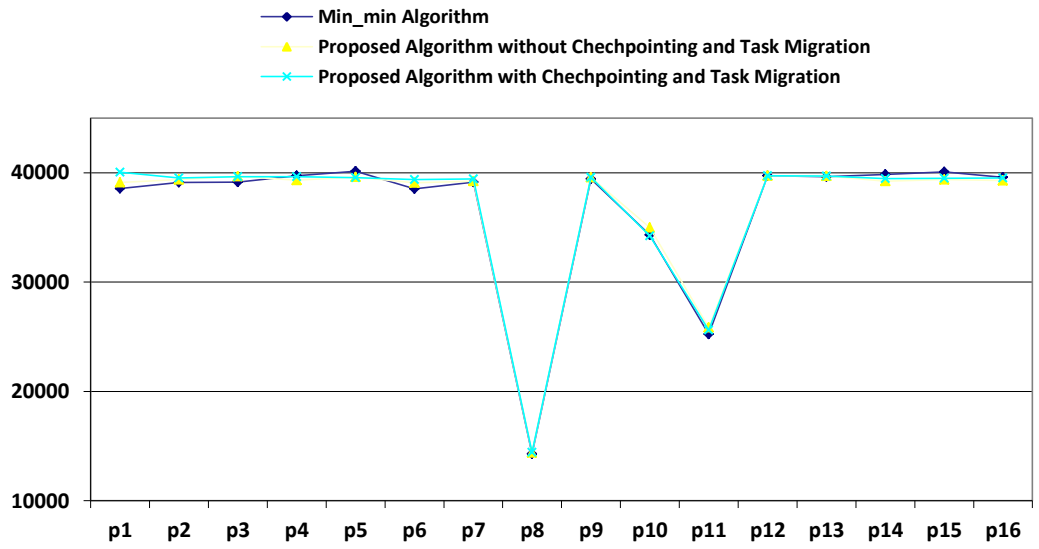

Fig5. Comparison of result obtained by Min-min algorithm, proposed algorithm without checkpointing and proposed algorithm with checkpointing for 20% fault occurrence rate based on table 2

Table 3. The used parameters for simulating of proposed algorithm with task migration and Min-min algorithm

Number of tasks	512
Number of machines	16
Task computational requirements	5 - 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	0
Delay Rate	0, 5, 10, 15 and 20 (%)

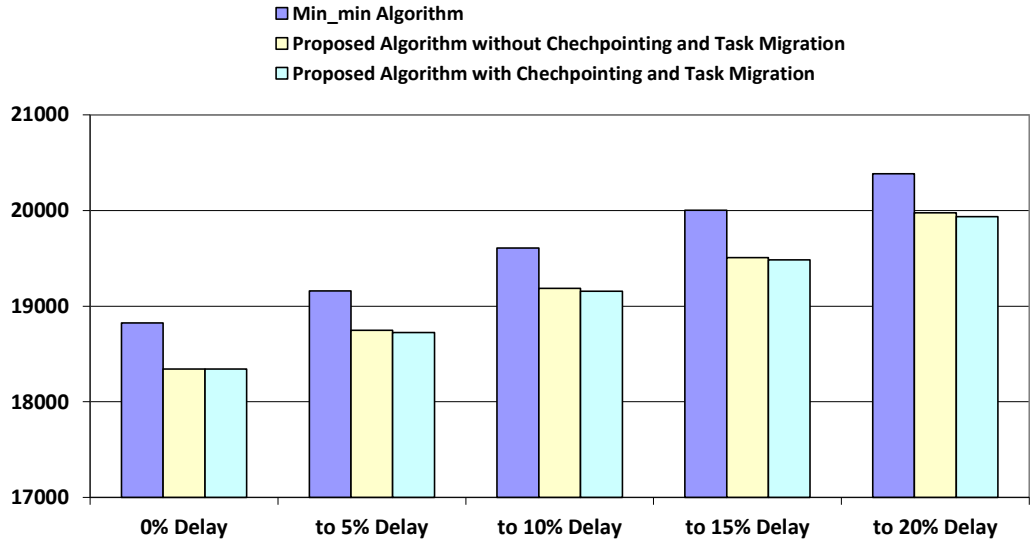

Fig6. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for various delay rates based on table 3

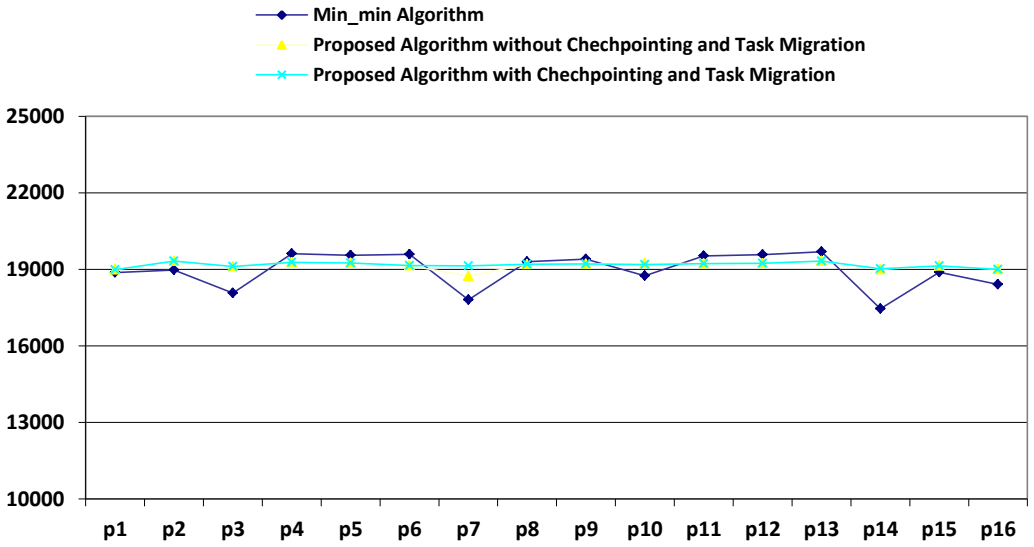

Fig7. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 10% delay rate based on table 3

Table 4. The used parameters for simulating of proposed algorithm with task migration and Min-min algorithm

Number of tasks	1024
Number of machines	16
Task computational requirements	5 - 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	0
Delay Rate	0, 5, 10, 15 and 20 (%)

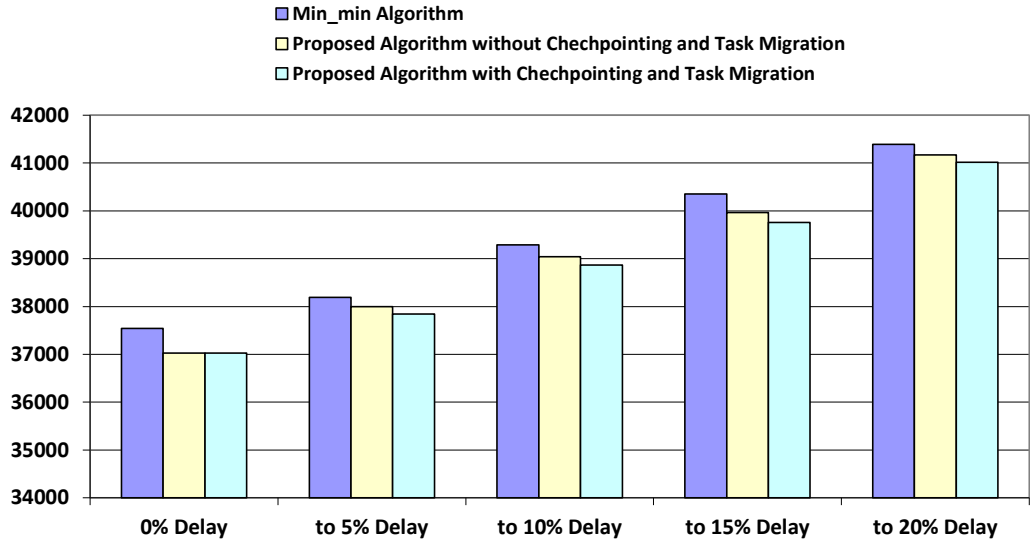

Fig8. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for various delay rates based on table 4

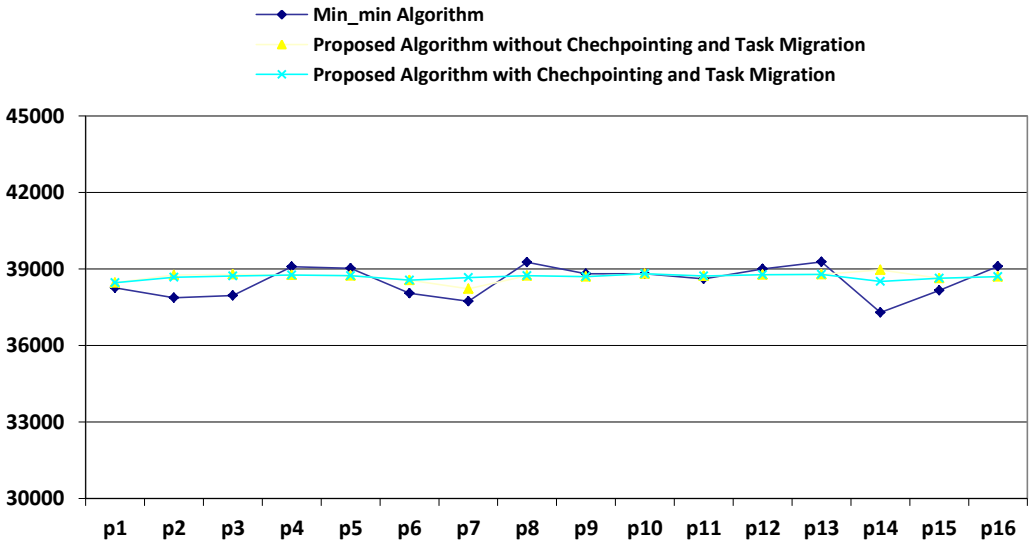

Fig9. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 10% delay rate based on table 4

Table 5. The used parameters for simulating of proposed algorithm with fault tolerance and task migration and Min-min algorithm

Number of tasks	512
Number of machines	16
Task computational requirements	5 - 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	20 (%)
Delay Rate	10 (%)

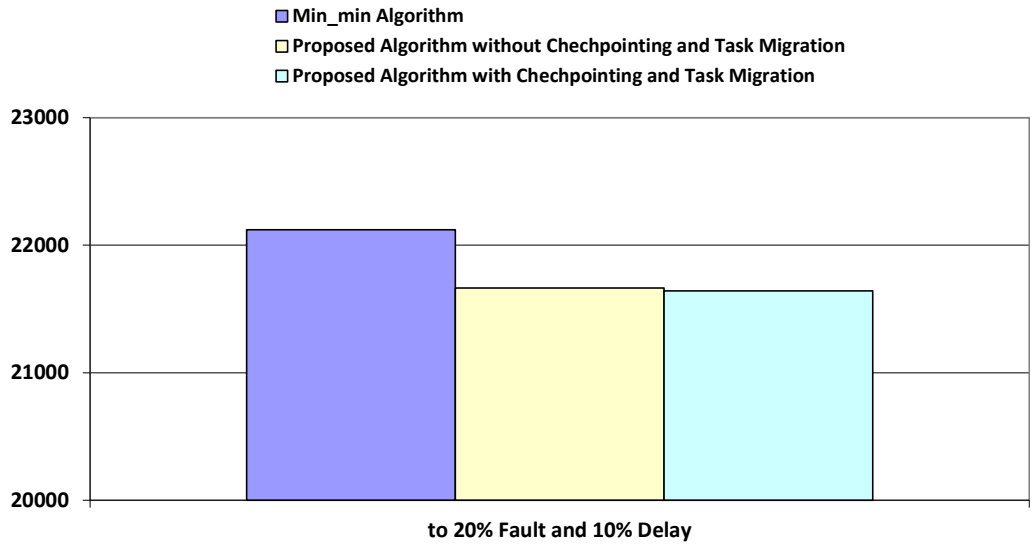

Fig10. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 20% fault occurrence rate and 10% delay rate based on table 5

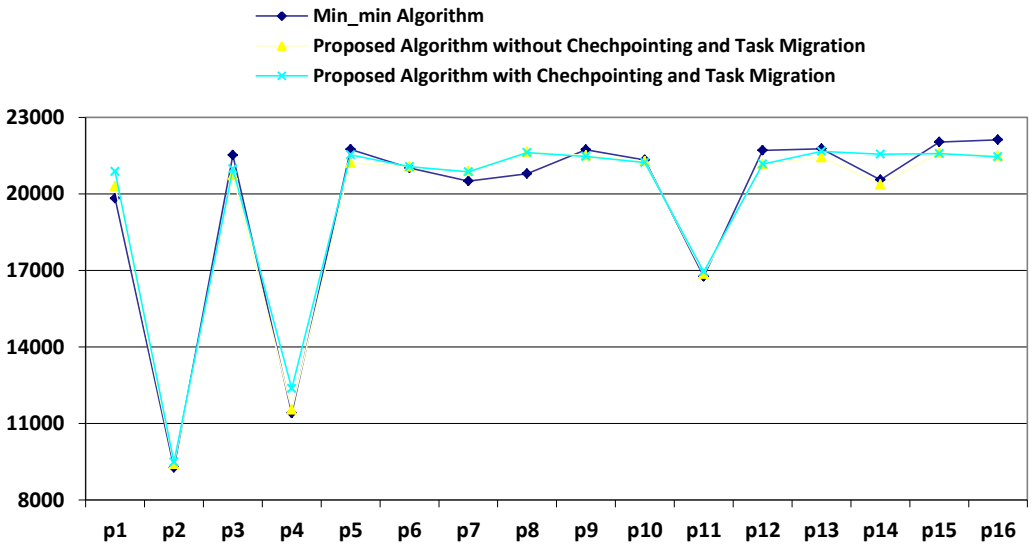

Fig11. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 20% fault occurrence rate and 10% delay rate based on table 5

Table 6. The used parameters for simulating of proposed algorithm with fault tolerance and task migration and Min-min algorithm

Number of tasks	1024
Number of machines	16
Task computational requirements	5 - 50 (billion machine instructions)
Machine capabilities	10-100(million machine instructions)
Task send/ receive file size	0.1-100(Mb)
Network bandwidth	100-1000 (Mbps)
Failure coefficient	20 (%)
Delay Rate	10 (%)

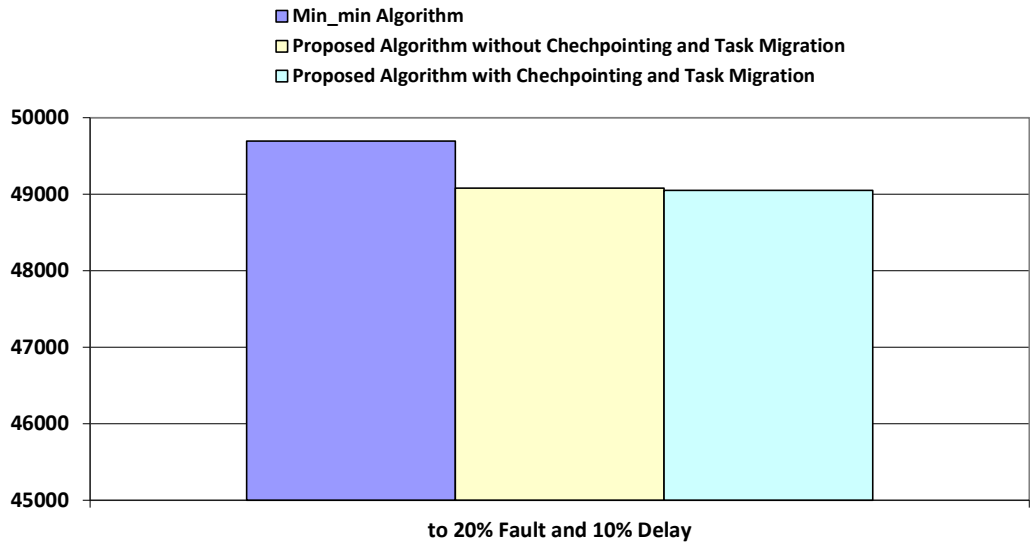

Fig12. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 20% fault occurrence rate and 10% delay rate based on table 6

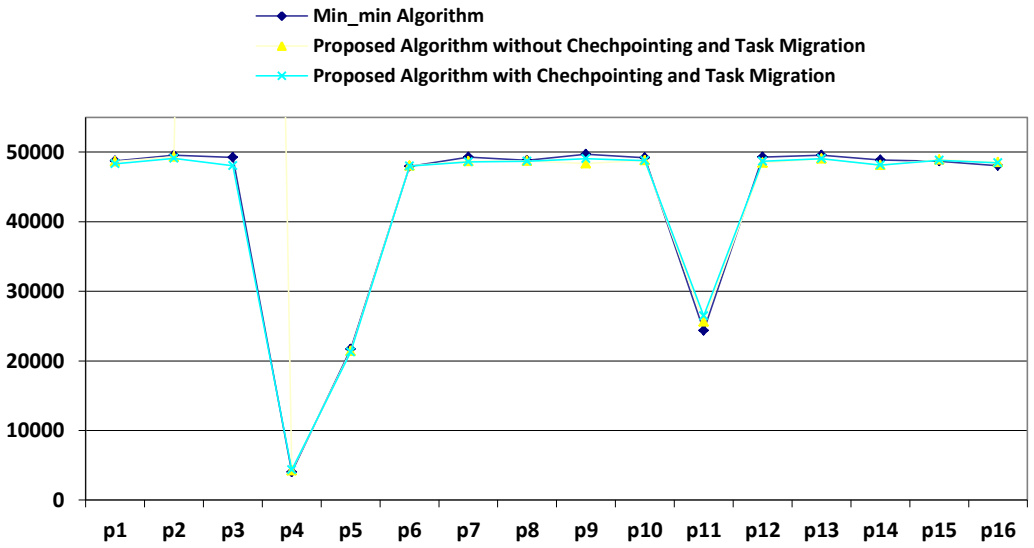

Fig13. Comparison of result obtained by Min-min algorithm, proposed algorithm without task migration and proposed algorithm with task migration for 20% fault occurrence rate and 10% delay rate based on table 6

6. Conclusion and Future Work

Experimental results show the proposed algorithm has better improvement than Min-min algorithm and reduces the time, cost and changes reliability to the best possible amount. With taking the fault tolerance with checkpointing method, task migration for tasks with long delay and priority for mapping tasks in simulated environment, we will achieve more efficiently in proposed scheduling algorithm performance, throughput maximization and reduced measure of the throughput of grid computing systems. The future research will be focused on communication cost, other delay times and also consider the nodes and parameters to be dynamic regarding the environmental conditions in grid systems.

7. References

[1] I. Foster and C. Kesselman, Eds., "the Grid: Blueprint for a Future Computing Infrastructure". Morgan Kaufmann Publishers, 1999

[2] 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.

[3] A. Ghorbannia Delavar , A. R. Khalili Boroujeni and J. Bayrampoor, International Journal of Computer Science Issues, "BPISG: A Batching Heuristic Scheduling Algorithm With Taking Index Parameters for Mapping Independent Tasks on Heterogeneous Computing Environment", Vol. 8, Issue 6, No 1, November 2011.

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

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

[6] 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.

[7] Thilo Kielmann, Vrije Universiteit, Amsterdam, "Scalability in Grid". PPT Core GRID, Bridging Global Computing with Grid (BIGG), Nov. 29, 2006.

[8] Malarvizhi Nandagopal, V. Rhymend Uthariaraj, International Journal of Engineering Science and Technology "Fault Tolerant Scheduling Strategy for Computational Grid Environment" Vol. 2(9), 2010, 4361-4372.

[9] B. Yagoubi , Department of Computer Science, Faculty of Sciences, University of Oran and Y. Slimani , Department of Computer Science, Faculty of Sciences of Tunis, "Task Load Balancing Strategy for Grid Computing".

[10] J. Jayabharathy, and Ayeshaa Parveen, International Journal of Recent Trends in Engineering, "A Fault Tolerant Load Balancing Model for Grid Envirnment" Vol. 2(9), 2009.

[11] Yuan-Jin Wen and Sheng-De Wang, "Minimizing Migration on Grid Environments: an Experience on Sun Grid Engine", Journal of Information Technology and Applications, Vol. 1, No 4, March 2007, 297-304.

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

[13] Jia Yu and Rajkumar Buyya, " Workflow Scheduling Algorithms for Grid Computing ",Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne.

[14] Hesam Izakian, Ajith Abraham, Senior Member, IEEE, Václav Snášel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments".

[15] Kamaljit Kaur, Amit Chhabra, Gurvinder Singh, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), Volume (4).

[16] Cong Du, Xian-He Sun and Ming Wu, "Dynamic Scheduling with Process Migration", National science Foundation.

[17] P.Kokkinos, K. Christodoulopoulos, A. Kretsis and E. Varvarigos, "Data Consolidation: A Task Scheduling and Data Migration Technique for Grid Networks", IEEE Computer Society, 2008.

[18] Abderezak Touzene, Sultan Al-Yahai, Hussien AlMuqbali, Abdelmadjid Bouabdallah and Yacine Challal, "Performance Evaluation of Load Balancing in Hierarchical Architecture for Grid Computing Service Middleware", International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[19] Belabbas Yagoubi, "Load Balancing Strategy in Grid Environment", Journal of International Technology and Applications, Vol. 1, No 4, March 2007, 285-296.

[20] Ashish Revar, Malay Andhariya, Dharmendra Sutariya, "Load Balancing in Grid Environment using Machine Learning-Innovative Approach", Journal of International Technology and Applications, Vol. 8, No 10, October 2010.

AUTHOR'S PROFILE

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, Payame Noor University, Tehran, IRAN. He is also the Director of Virtual University and Multimedia Training Department of Payame 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 Payame 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 Payame Noor University, Tehran, IRAN. His research interests include computer networks, grid and scheduling algorithm.