Heuristic Based Task Scheduling In Grid

Kamali Gupta [1], Manpreet Singh[2]
1Department of Computer Engineering, GIMT, Kanipala,
Kurukshetra, Haryana,, India kamaligupta@gimtkkr.com
2Department of Computer Engineering, M.M. University,
Mullana, Ambala, Haryana,, India
drmanpreetsinghin@gmail.com

Abstract— **Grid computing is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. Efficient scheduling of complex applications in a grid environment reveals several challenges due to its high heterogeneity, dynamic behavior and space shared utilization. Objectives of scheduling algorithms are increase in system throughput, efficiency and reduction in task completion time. The main focus of this paper is to highlight the merits of resource and task selection technique based on certain heuristics.**

Keyword- **Grid Computing, Task Scheduling, Min-Min, Max-Min, Suffrage, Makespan.**

I. Introduction

The term grid [1] is increasingly appearing in computer literature, generally referring to some form of system framework into which hardware or software components can be plugged and which permits easy configuration and creation of new functionality from existing components. Grids enable the sharing, selection and aggregation of a wide variety of resources including supercomputers, storage systems, data sources and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering and commerce [2]. The computing power of grid is aggregated by that of various organizational or individual computing resources and grid users need only to submit computational tasks to it. There are still some difficult issues impeding the development of grid, among which is the issue of grid task scheduling [3][4]. In order to efficiently utilize available grid resources and promptly complete tasks assigned to the grid, providing a suitable task scheduling strategy for the grid computing is necessary [5][6].

The objective of this research work is to make a comparison among various heuristic based scheduling algorithms under different resource/task mapping environments. In Min-Min algorithm [7], the smaller tasks are chosen first, making use of resources with high computational power. As a result, the schedule prepared by Min-Min is not optimal when number of smaller tasks exceeds the larger one. Max-Min algorithm [9] schedules larger tasks first. But in some cases, the makespan may increase due to the execution of larger tasks first. The rationale behind Suffrage [10] is that a task should be assigned to a certain resource and if it does not go to that resource, it will suffer the most. For each task, its suffrage value is defined as the difference between its best Minimum Completion Time (MCT) and its second-best MCT. Tasks with high suffrage value take precedence during scheduling.

II HEURISTIC BASED SCHEDULING ALGORITHMS

The resource selection process is used to choose one or more resources from the list of candidates for a given resource requirement. Since all resources in the list could meet the minimum requirements imposed by the task, so an algorithm is needed that can choose the best resource for executing the task.

Min-Min Algorithm: Min-Min [8] begins with the set MT (Meta Task) of all unassigned tasks and has two phases. In the first phase, the set of minimum expected completion time for each task in MT is found. In the second phase, the task with the overall minimum expected completion time from MT is chosen and assigned to the corresponding macine. Then this task is removed from MT and the process is repeated until all tasks in the MT are mapped as shown in Fig. 1. However, the Min- Min algorithm is unable to balance the load well as it usually does the scheduling of small tasks initially.

BEGIN

1. *While (J != Null) // J is set of jobs*
2. *For each job j_i £ J*
   For each machine m_j

Calculate the completion time

C_{ij} = E_{ij} + R_j // C_{ij}, E_{ij} and R_j represents completion time, execution time and ready time of job j_i on machine m_j

End For

End For

3. *For each job j_i £ J*
   Find the minimum completion time and the machine that obtains it.
   End For
4. *Search the job j_u having minimum completion time among all unassigned jobs.*
5. *Allocate j_u to machine m_v that has resulted in obtaining minimum completion time of j_u.*
6. *Delete job j_u from the job set J: J = J - j_u*
7. *Update the ready time of machine m_v as: $R_v = C_{uv}$*
   End While

END

Fig. 1: The Min-Min Heuristic

Max-Min Algorithm: Max-Min [10] differs from Min-Min in second phase, where tasks with overall maximum expected completion time from MT is chosen and assigned to corresponding machine as shown in Fig. 2.
In other words, Min-Min gives priority to the task that has the shortest earliest completion time, whereas at the time of each scheduling instance, Max-Min tends to schedule the longer task first.

BEGIN

1. *While (J != Null)*
2. *For each job j_i £ J*
   For each machine m_j
   Calculate the completion time
   C_{ij} = E_{ij} + R_j // C_{ij}, E_{ij} and R_j represents completion time, execution time and ready time of job j_i on machine m_j
   End For
   End For
3. *For each job j_i £ J*
   Find the minimum completion time and the machine that obtains it.
   End For
4. *Search the job j_u having maximum completion time among all unassigned jobs.*
5. *Allocate j_u to machine m_v that has resulted in obtaining maximum completion time of j_u.*
6. *Delete job j_u from the job set J: J = J - j_u*
7. *Update the ready time of machine m_v as: $R_v = C_{uv}$*
   End While

END

Fig. 2: The Max-Min Heuristic

Switcher Algorithm: Switcher [12] selects between the Max-Min and Min-Min algorithm on the basis of Standard Deviation (SD) of minimum completion time of unassigned jobs. As the name depicts, it switches between the two algorithms selecting the best between the two, while making each scheduling decision. A position in the list of unassigned jobs where the difference in completion time between the two successive jobs is more than the value of SD is searched. If it lies in first half of the list, then Min-Min algorithm is evaluated as the number of longer jobs is more, otherwise Max-Min is evaluated by taking the last job from the list. If this position does not exist, then SD is compared with a threshold value. Allocation of job to a machine is implemented using Min-Min strategy, if SD is smaller than threshold value. Otherwise, Max-Min is selected for assigning the next job as shown in Fig. 3.

BEGIN

1. *While (J != Null)*
2. *For each job j_i £ J*
   For each machine m_j
   Calculate the completion time
   C_{ij} = E_{ij} + R_j // C_{ij}, E_{ij} and R_j represents completion time, execution time and ready time of job j_i on machine m_j
   End For
   End For
3. *For each job j_i £ J*
   Find the minimum completion time and the machine that obtains it.
   End For

4. *Calculate the SD of completion time of all unassigned jobs.*
5. *Sort all unassigned jobs in increasing order of their completion times.*
6. *Find a position in this list where difference in completion time of two consecutive jobs is more than SD.*
7. *If this position is in the 1st half of list of unassigned jobs or SD< threshold value*

   Apply min-min heuristic

   Else

   Apply max-min heuristic

   End While

END

Fig. 3: The Switcher Heuristic

Suffrage Algorithm: In Suffrage [11], the minimum and second minimum completion time for each job are found in first step. The difference between these two values is defined as suffrage value as shown in Fig. 4. In second step, the task with maximum suffrage value is assigned to corresponding machine with minimum completion time.

BEGIN

1. *While (J != Null)*
2. *For each job* j_i *£ J*

   For each machine m_j

   Calculate the completion time

   $C_{ij} = E_{ij} + R_j$ *//* C_{ij}, E_{ij} *and* R_j *represents completion time, execution time and ready time of job* j_i *on machine* m_j

   End For

   End For
3. *For each job* j_i *£ J*

   (a) *Find the First minimum completion time* (FST_MCT_i) *and second minimum completion time* (SEC_MCT_i) *of job* j_i.

   (b) *Calculate the suffrage value:* $SV_i = SEC\_MCT_i - FST\_MCT_i$

   End For
4. *Search the job* j_u *having maximum suffrage value among all unassigned jobs.*
5. *Allocate* j_u *to machine* m_v *that has resulted in obtaining minimum completion time of* j_u.
6. *Delete job* j_u *from the job set J:* $J = J - j_u$
7. *Update the ready time of machine* m_v *as:* $R_v = C_{uv}$

   End While

END

Fig. 4: The Suffrage Heuristic

III AN ILLUSTRATION OF SCHEDULING ALGORITHMS

This section presents the generated schedule for various task scheduling algorithms with the help of an example. Consider the Expected Time to Compute (ETC) matrix as represented in Table 1. The table shows the expected execution times of 10 unassigned jobs on 5 machines. X denotes that the machine does not have capability to execute that particular job.

Table 1 ETC Matrix of Unassigned Tasks

Parameters	m0 (Machine)	m1	m2	m3	m4
J_0 (Task)	X	29	16.8	X	X
J_1	12.7	X	38.5	34.3	9.5
J_2	36.3	19.4	22	X	17.6
J_3	X	X	X	26.7	23.2
J_4	X	7.8	X	X	32.7
J_5	X	35.5	X	30.8	8.1
J_6	31.4	20.9	X	X	37.9
J_7	X	5	X	23.8	23.2
J_8	27.4	36.8	39.9	X	22.7
J_9	37.5	8.9	26.4	X	12.5

Fig. 5 shows the result for Min-Min with makespan = 53.10 using above specified predicted execution times of unassigned tasks.

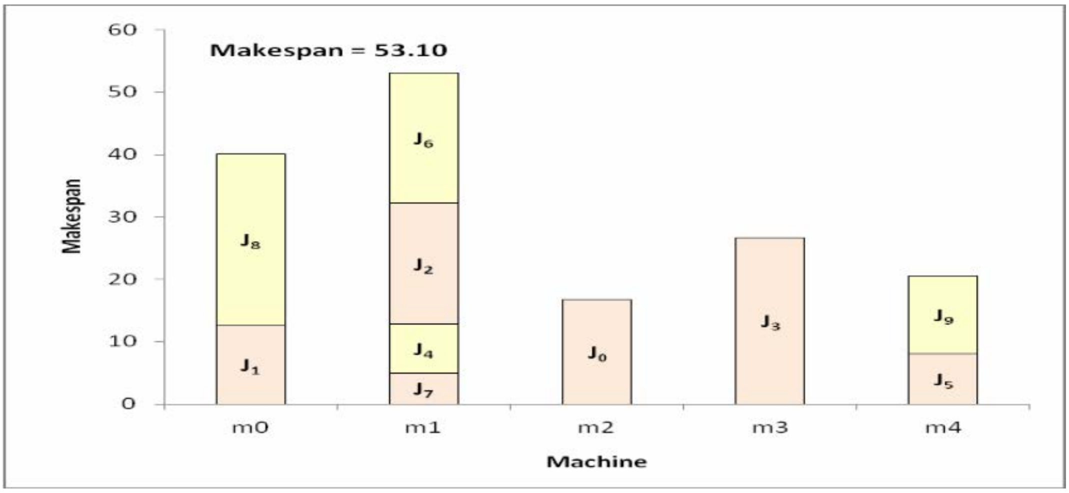

Fig. 5: Makespan Under Min-Min Algorithm

After applying the Max-Min algorithm, the generated schedule is presented in Fig. 6 with makespan = 62.70 .

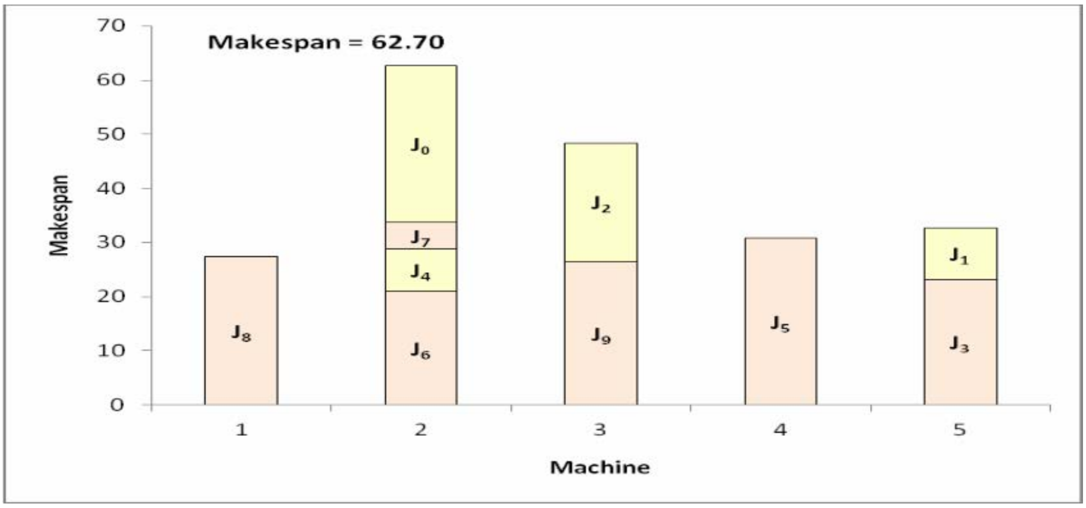

Fig. 6: Makespan Under Max-Min Algorithm

Fig. 7 illustrates the schedule generated by Switcher algorithm having makespan = 53.10 .

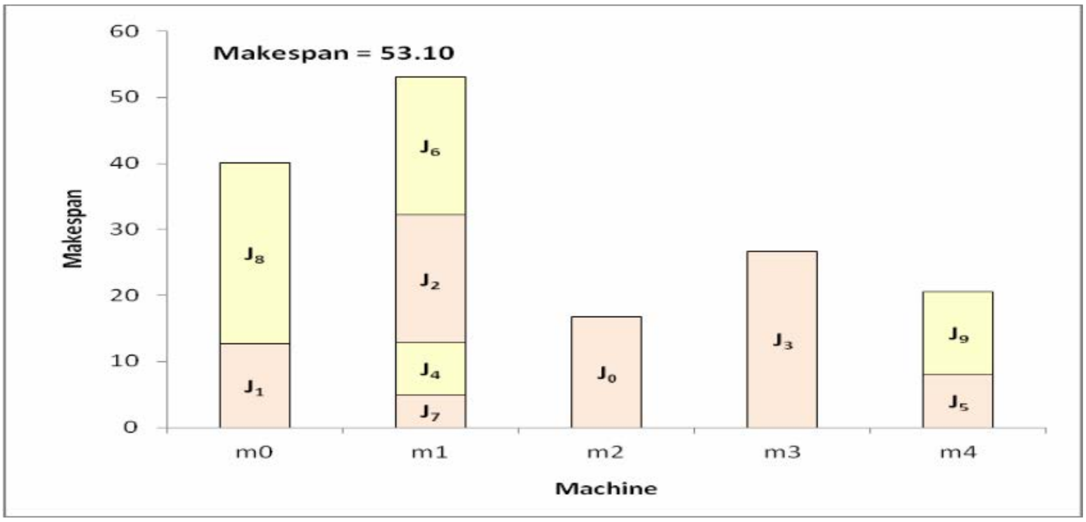

Fig. 7: Makespan Under Switcher Algorithm

Fig. 8 presents the resulted schedule for Suffrage algorithm with makespan = 42.60.

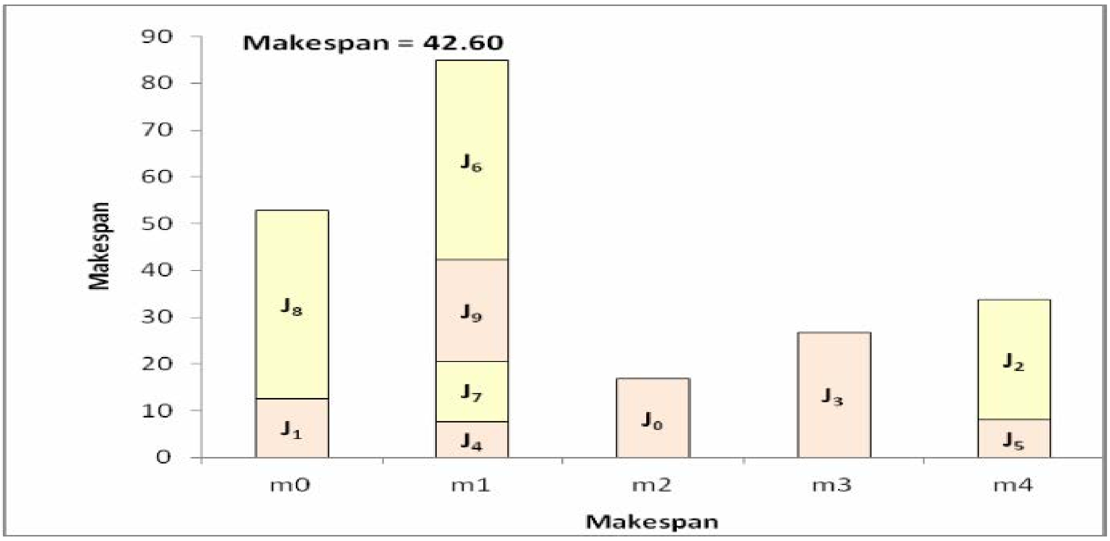

Fig. 8: Makespan Under Suffrage Algorithm

IV SIMULATION RESULTS & DISCUSSION

This section presents the comparison among various task scheduling algorithms. The functional code is implemented using simulator built in Java on an Intel core 2 duo, 2 GHz window based laptop to evaluate the performance of various scheduling algorithms under different load conditions in terms of variation in number of tasks as shown in Table 2 (Fig. 9), Table 3 (Fig. 10), Table 4 (Fig. 11) and Table 5 (Fig. 12).

Scenario 1: Systems having low load

Table 2 Performance Under Low Load Conditions

No. of Simulation Runs = 10			
No. of Task= 30			
No. of Machines = 10			
Min-Min	Max-Min	Switcher	Suffrage
46	59.3	51.2	38.9

Fig. 9: Makespan Under Low Load Conditions

Scenario 2: Systems having medium load

Table 3 Performance Under Medium Load Conditions

No. of Simulation Runs = 10			
No. of Task = 70			
No. of Machines = 10			
Min-Min	Max-Min	Switcher	Suffrage
95.1	130.1	90.9	79.5

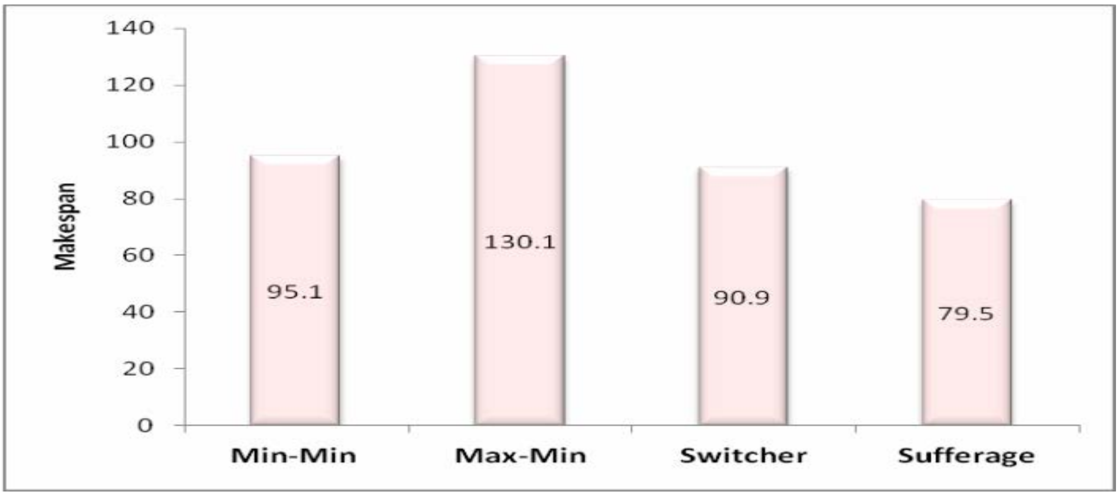

Fig. 10: Makespan Under Medium Load Conditions

Scenario 3: System having high load

Table 4 Performance Under High Load Conditions

No. of Simulation Runs = 10			
No. of Task = 120			
No. of Machines = 10			
Min-Min	Max-Min	Switcher	Suffrage
159.7	248.1	159.7	152.5

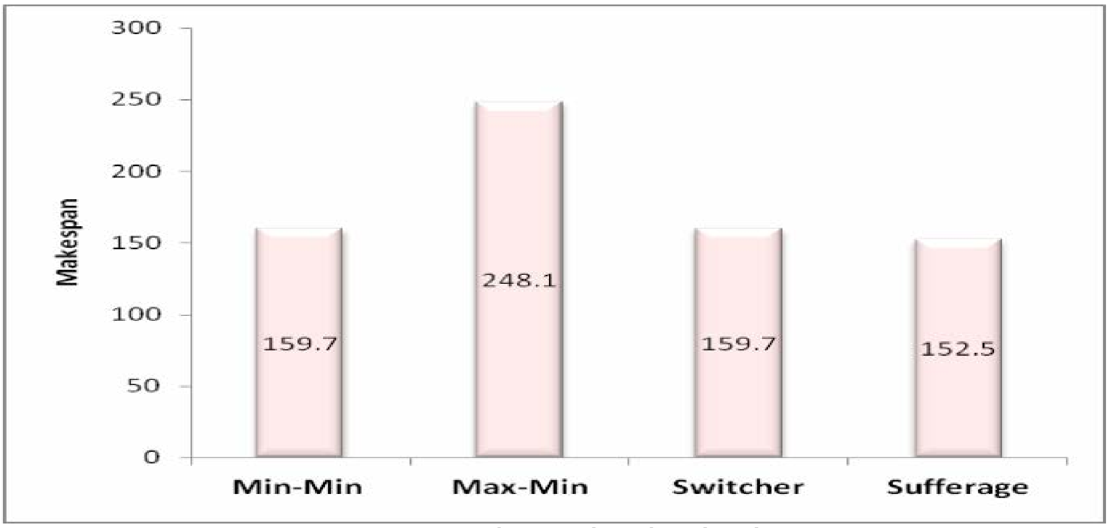

Fig. 11: Makespan Under High Load Conditions

Comparison among different heuristics under various load conditions is shown in Fig. 12.

Table 5 Performance Under Different Load Conditions

No. of Task	Min-Min	Max-Min	Switcher	Suffrage
25	34.5	34.6	38	28.2
50	59.8	81.5	59.8	52.1
75	67.3	122.7	67.3	65.2
100	108.7	144.5	97.2	93.7

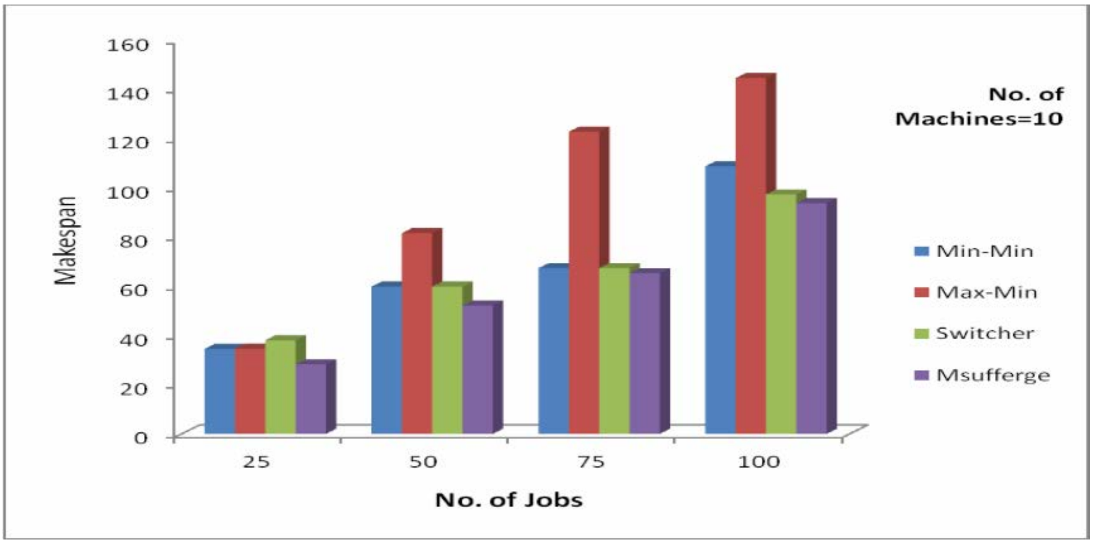

Fig. 12: Makespan of Different Heuristics Under Various Load Conditions

V CONCLUSION

In this paper, the working of various task scheduling algorithms is presented. The main goal of task scheduling is to reduce the overall makespan of the jobs submitted in the grid. When it gets minimized, the performance of entire grid gets optimized automatically. A performance comparison among various scheduling heuristics has been made under various load conditions in terms of variation in number of tasks and task length. The heuristic with shortest makespan is declared the best to perform task scheduling in grid. Simulation result shows that the Suffrage scheduling algorithm generates an optimum schedule and outperforms the other conventional algorithms.

REFERENCES

[1] I. Foster, and C. Kesselman, *The Grid: Blueprint for a new Computing Infrastructure,* 2nd ed.: Morgan Kauffman publishers, 2004.

[2] R. Buyya and S. Venugopal, "A Gentle Introduction to Grid Computing and Technologies" *CSI Communication*, pp. 9-19, July 2005.

[3] Z. Jinnquan, N. Lina and J. Changjun, "A Heuristic Scheduling Strategy for Independent Tasks on Grid", in *Proc. 8th Inter. Conf. on High-Performance Computing in Asia-Pacific Region (HPCASIA'05),* 2005, pp. 588-593.

[4] H. Zhang, C. Wu, Q. Xiong, L. Wu and G. Ye, "Research on an Effective Mechanism of Task-scheduling in Grid Environment", in *Proc. 5th Inter. Conf. on Grid and Cooperative Computing*, 2006, pp. 86-92.

[5] H. Baghban and M. Rahmani, "A Heuristic on Job Scheduling in Grid Computing Environment", in *Proc. 7th Inter. Conf. on Grid and Cooperative Computing (GCC'08),* 2008, pp. 141-146.

[6] G. Hong-cui1, Y. Jiong, H. Yong and L. Hong-wei, "User QoS and System Index Guided Task Scheduling in Grid Computing", in *Proc. 3rd China Grid Annual Conference,* 2008, pp. 109-112.

[7] T. Kokilavani and D. Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", *International Journal of Computer Applications*, Vol.20, No.2, pp 43-49, 2011.

[8] M. Wu, W. Shu and H. Zhang, "Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing System", in *Proc. 9th Heterogeneous Workshop (HCW'00),* 2000, pp. 375-385.

[9] E. Munir, J. Li , S. Shi and Q. Rasool, " Performance Analysis of Task Scheduling Heuristics in Grid", in *Proc. 6th Inter. Conf. on Machine Learning and Cybernetics*, 2007, pp. 3093-3098.

[10] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent tasks onto Heterogeneous Computing Systems," *Journal of Parallel and Distributed Computing*, Vol. 59, No. 2, pp. 107-131,1999.

[11] E. U. Munir, J. Li and S. Shi, "QoS Sufferage Heuristic for Independent Task Scheduling in Grid," *Information Technology Journal*, Vol. 6, No. 8, pp. 1166-1170, 2007.

[12] M. Singh andP.K.Suri, "QPS$_{\text{Max-Min<>Min-Min}}$: A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid", *Information Technology Journa,l* Vol. 7, No. 8, pp. 1176-1181, 2008.