TLLB: Two-Level Load Balanced Algorithm for Static Meta-Task Scheduling in Grid Computing

S. Vaaheedha Kfatheen
Research Scholar,
Bharathiar University, Coimbatore,
Tamil Nadu, India.

M. Nazreen Banu, Ph.D.
Professor, Dept of MCA,
M.A.M. College of Engg,
Tiruchirappalli, TN, India.

S. Kavi Selvi
Research Scholar, Jamal
Mohamed College,
Tiruchirappalli, TN, India.

ABSTRACT
Doing computation on the collection of computer resources from multiple locations to reach a common goal is knows as grid computing. Task scheduling is very important problem in complex grid environment. Prior, there are numerous number of algorithms were proposed to do effective task scheduling. Among them the min-min algorithm is simple and well known scheduling algorithm. Even it works efficiently, some drawbacks in this with respect of load balancing and in resource utilization. To overcome these drawbacks, a new Two Level Load Balanced (TLLB) grid scheduler algorithm is proposed. In First Level min-min algorithm is used to create ITQ and in Second Level a new Transformation technique is used to reschedule. The performance analyses show that the proposed algorithm improves the performance in both make span and effective utilization of resources.

Keywords
Grid computing, Min-min, Load balancing, resource utilization, Task Scheduling, Flow-time

1. INTRODUCTION
As mentioned in abstract collection of computer resources from multiple locations to reach a common goal is known as Grid computing. Grid computing is distinguished from conventional high performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application. Grid computers also tend to be more heterogeneous and geographically detached than cluster computers. Task scheduling in a grid environment is a main issue. Grid resource management involves dealing with three classes of stakeholders - end users, owners of resources, and grid administrators. Each class of stakeholders has their own perspective and preferences, which result in different, often contradictory, criteria for scheduling. To increase the level of satisfaction of these stakeholders grid management system must be used the scheduling heuristic.

Scheduling [1] is considered to be an important issue in the current Grid scenario. The demand for effective scheduling increases to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation which minimizes the schedule length of jobs and effectively utilize the resources. The three main phases [2] of grid scheduling are resource discovery, gathering resource information and job execution. The choice of the best pair of jobs and resources in the second phase has been proved to be NP-complete problem.

Large numbers of task scheduling algorithms are available to minimize the make span [3], [4], [5], [6], [7], [8]. All these algorithms try to find resources to be allocated to the tasks which will minimize the overall completion time of the jobs. Minimizing overall completion time of the tasks does not mean that it minimizes the actual execution time of individual task.

Two simple well-known algorithms used for grid scheduling are Min-Min and Max-min [9], [3], [5], [6], [10], [8]. These two algorithms work by considering the execution and completion time of each task on the each available grid resource.

This paper is proposed to rectify the limitation of Min-Min algorithm and tries to bring a new algorithm which gives reduced make span and high resource utilization.

The remainder of this paper is structured as follows: Section 2 presents the related works and several well known scheduling algorithms which are used as benchmarks of many other works. In Section 3 the concept of task scheduling in grid environments is introduced, In Section 4, a new scheduling algorithm is proposed. Section 5 compares the scheduling algorithms and presents the results of the comparison. Finally, Section 6 presents concluding remarks and future work.

2. RELATED WORKS
As discussed before the Min-Min and the Max-Min algorithms are simple and researched by maximum number of research scholars. The previous works that were related to this proposal is analyzed here.

The Min-Min algorithm first is used to find the minimum execution time of all tasks. Then it is used to choose the task with the least execution time among all the tasks. The algorithm is being preceded by assigning the task to the resource that produces the minimum completion time. The same procedure is repeated by Min-Min until all tasks are scheduled.

The limitation of Min-Min algorithm is that it chooses smaller tasks first which makes use of resource with high computational power. As a result, the schedule produced by Min-Min is not optimal when number of smaller tasks exceeds the large ones. To overcome this difficulty [11], Max-min algorithm schedules larger tasks first. But in some cases, the make span may increase due to the execution of larger tasks first. The waiting time of smaller tasks is also increased in Max-min.

Braun et al have studied the relative performance of eleven heuristic algorithms for task scheduling in grid computing [9]. They have also provided a simulation basis for researchers to test the algorithms. Their results show that Genetic Algorithm (GA) performs well in most of the scenarios and the relatively simple Min-Min algorithm performs next to GA and the rate of improvement is also very small. The simple algorithms proposed by Braun are Opportunistic Load Balancing (OLB), Minimum Execution Time (MET), Minimum Completion Time (MCT), Min-Min, Max-min.

Opportunistic Load Balancing (OLB) assigns the jobs in a random order in the next available resource without considering the execution time of the jobs on those resources. Thus it provides a load balanced schedule but it produces a very poor make span.

Minimum Execution Time (MET) assigns jobs to the resources based on their minimum expected execution time without considering the availability of the resource and its current load. This algorithm improves the make span to some extent but it causes a severe load imbalance.

Minimum Completion Time (MCT) assigns jobs to the resources based on their minimum completion time. The completion time is calculated by adding the expected execution time of a job on that resource with the resource's ready time. The resource with the minimum completion time for that particular job is selected. But this algorithm considers the job only one at a time.

Switching Algorithm (SA) is heuristic of scheduling combines the best features of MCT and MET methods of scheduling. Method was tried to use better load balancing of MCT and execution on fastest resource of MET. Here the idea was to first use the MCT till a threshold of balance is reached followed by MET. The load unbalance by assigning tasks on faster resources was created by MET. Here MCT and MET are used in cyclic manner [10], [12]. Problem is solvable using OLB in O(nm) time.

Work Queue (WQ) Work Queue is a very simple heuristic of task allocation. Tasks are randomly selected from the list of unassigned tasks and assigned to the resource with minimum workload. Task assignment repeated in similar manner till list of unassigned tasks gets exhausted [13].

Min-Min algorithm starts with a set of all unmapped tasks. The resource that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. Compared to MCT this algorithm considers all jobs at a time. So it produces a better make span.

Max-Min is similar to Min-Min algorithm. The resource that has the minimum completion time for all jobs is selected. Then the job with the overall maximum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. The idea of this algorithm is to reduce the wait time of the large jobs.

LJFR-SJFR Largest Job (task) on Fastest Resource – Shortest Job (task) on Fastest Resource (LJFR-SJFR) method allocates largest task to fastest resource to reduce the make span and allocates smallest task to fastest resource to reduce the flow time of the schedule [14] and [15]. Heuristic LJFR-SJFR can solve problem in O(n2m) time.

Doreen. D et al., [16] have proposed an efficient Set Pair Analysis (SPA) based task scheduling algorithm named Double Min-Min Algorithm in which scheduling was performed in order to enhance system performance in hyper cubic P2P Grid (HPGRID). The simulation result were shown that the SPA based Double Min-Min scheduling minimizes the make span with load balancing and the high system availability is guaranteed in system performance.

Among all the algorithms stated the Min-Min algorithm is simple and fast, at the same time it produces a better make span. But it considers the shortest jobs first so it fails to utilize the resources efficiently which leads to a load imbalance.

This work is proposed to overcome the drawback of the Min-Min algorithm. Two-Level Min-Min algorithm is proposed which improves the load balancing as well as and produces a make span better than the Min-Min algorithm.

3. PROBLEM DEFINITION

The effective scheduling algorithm must to minimize the make span and should utilize all the available resources. Using the ETC matrix model, the scheduling problem can be defined as follows:

Let task set T = t1, t2, t3, …. , tn be the group of tasks submitted to scheduler. As we discussed before tasks are independent not having any relationship with in them.

Let Resource set R = m1, m2, m3... mk be the set of resources.

The ETC [Ti, Rj] contains the matrix of finishing time of ith task (ti) on the jth resource (rj).

Min(ETC[Ti,Rj) is the minimum of Earliest completion time of task ti on resource list rj where j varies from 0 to k

CT[ti] is the list completion times of ti where i varies from 0 to n.

Make span MS is calculated as MAX(CT[ti]) V i =>0 to n

4. PROPOSED ALGORITHM

To avoid the drawbacks of the Min-Min algorithm many improved algorithms have been proposed in the literature. All the problems discussed in those methods are taken and analyzed to give a more effective schedule. The algorithm proposed in this paper outperforms all those algorithms both in terms of make span, resource utilization, flow time and load balancing. Thus a better load balancing is achieved and the total response time of the grid system is improved. The proposed algorithm applies the Min-Min strategy in the first level and then reschedules by transferring the tasks from maximum loaded resource to minimum loaded resource.

Proposed Two-Level Load Balanced grid scheduler algorithm (TLLB) has two levels. In the first level, Initial Tasks Queue (ITQ) is created using Min-Min strategy. The created ITQ is processed further to remove the load imbalance in second level. The load imbalance is removed by transfer tasks from maximally loaded resource to minimally loaded resource.

The penalty made for transfer of tasks from maximally loaded resource to minimally loaded resource should be minimum. And further the task completion time should not be much larger than the average resource completion time.

The algorithm is given below for the above explanations.

Algorithm 1: Algorithm for TLLB

1. let T as list of task t_i ∀ i = 0 to n
2. let R as list of Resource m_j ∀ j=0 to k
3. ETC[T_i, R_j]
4. ITQ= MINMIN (ETC[T_i, R_j])
5. FTQ=TRANSFORM_ITQ(ITQ)
6. For all t_i in FTQ

   ALLOT (t_i, m_j)
7. STOP

Where

ETC[Ti,Rj] is Earliest Task Completion matrix

ITQ means Initial Task Queue

FTQ means Final Transformed Queue

5. EXPERIMENTAL RESULTS

In this section, after the benchmark description, various scheduling algorithms were compared with the proposed algorithm TLLB. These algorithms are implemented using java environment and run on 12 different types of ETC matrices. For each algorithm and each type of ETC matrix, the results were averaged over 100 different ETC matrices of the same type (i.e., 100 mappings).

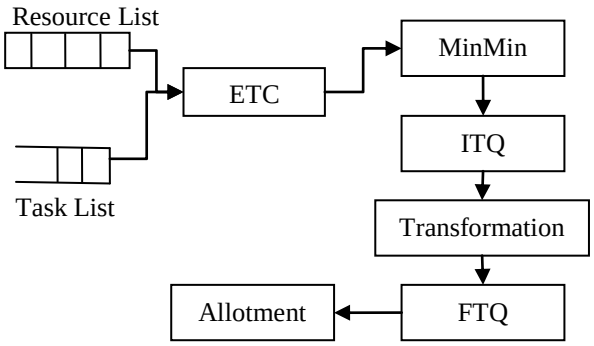

Fig 1: Architecture Diagram

5.1. Benchmark Description

In this paper, it is used the benchmark proposed by previous research scholars [17]. The simulation model is based on expected time to compute (ETC) matrix for 512 tasks and 16 resources. The instances of the benchmark are classified into 12 different types of ETC matrices according to the three following metrics: task heterogeneity, resource heterogeneity, and consistency. In ETC matrix, the amount of variance among the execution times of tasks for a given resource is defined as task heterogeneity. Resource heterogeneity represents the variation that is possible among the execution times for a given task across all the resources. Also an ETC matrix is said to be consistent whenever a resource Rj executes any task Ti faster than resource Rk ; in this case, resource Rj executes all tasks faster than resource Rk . In contrast, inconsistent matrices characterize the situation where resource Rj may be faster than resource Rk for some tasks and slower for others. Partially-consistent matrices are inconsistent matrices that include a consistent sub-matrix of a predefined size. Instances consist of 512 tasks and 16 resources and are labeled as u-x-yy-zz. The meaning of labels is given below,

u: Uniform distribution used in generating the matrices

x: Shows the type of inconsistency; c means consistent, i means inconsistent, and p means partially-consistent

yy: indicates the heterogeneity of the tasks; “hi” means high and “lo” means low

zz : represents the heterogeneity of the resources; “hi” means high and “lo” means low.

For example, “u-c-lohi” means low heterogeneity in tasks, high heterogeneity in resources, and consistent environment.

5.2. Make Span and Flow Time

The obtained make span and flow time using mentioned heuristics are compared in Tables 1 and 2 respectively. The results are obtained as an average of 100 simulations. Figure 2 and Figure 3 shows the geometric mean of make span and flow time for the 12 considered cases. Among most popular and extensively studied optimization criterion is the minimization of the make span. Small values of make span mean that the scheduler is providing good and efficient planning of tasks to resources. Another important optimization criterion is that of flow time, which refers to the response time to the user submissions of task executions. In general, the make span value is more important. As shown in Figure 2 and Figure 3, the new algorithm is better than all in make span value and second best in flow time value. Min-min gave the second best result (after new algorithm) in make span and best in flow time value. However the drawback of Min-min is that, it is unable to balance the load. The proposed algorithm retains the advantage of Min-min and reduces the idle time of the resources, which in turn leads to better make span.

Table 1. Comparison of Makespan values

Instant	Min-Min	Max-Min	MET	MCT	LJFR-SJFR	TLLB
u_c_hihi	8359675	11384672	37471299	10421624	11111938	7617740
u_c_hiho	140805.4	193054.6	1084093	174887.4	188252.4	147113.6
u_c_lohi	264837.4	381566.7	1352098	367303.6	387733.1	249359.1
u_c_lolo	4341.428	5845.362	37582.3	5260.055	5645.371	4370.133
u_i_hihi	3412919	7917378	4407507	4312583	6612596	3401023
u_i_hilo	78755.68	140923.8	94610.48	92855.91	119346.9	74654.02
u_i_lohi	109517.7	240528.8	174694.6	132816.1	202484.6	113021.8
u_i_lolo	1685.645	4077.709	2299.285	2037.35	3297.988	1562.64
u_p_hihi	5059343	9107811	24161058	6592924	8238410	5437803
u_p_hilo	93375.2	161822.7	594363.8	115587.6	143962.9	99232.1
u_p_lohi	119284.5	261085.7	653689.5	165151.3	225097.7	143003
u_p_lolo	2706.828	5132.242	19042.41	3336.118	4427.161	2728.391

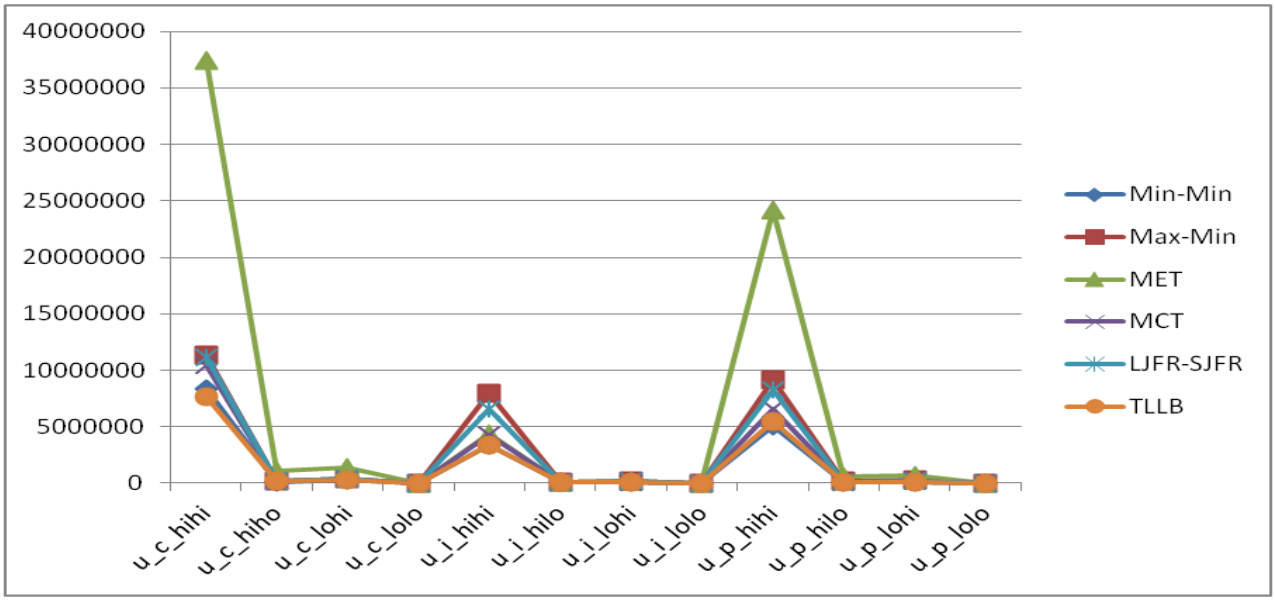

Fig 2: Graphical Representation of makespan values

Table 2. Comparison of Flowtime values

Instant	Min-Min	Max-Min	MET	MCT	LJFR-SJFR	TLLB
u_c_hihi	2.11E+07	3.12E+08	1.14E+08	2.80E+07	3.10E+07	1.94E+07
u_c_hiho	37299000	48799000	279999000	43999000	47999000	36599000
u_c_lohi	57199000	89999000	279999000	89499000	96699000	57499000
u_c_lolo	1339000	1759000	9929000	1559000	1669000	1319000
u_i_hihi	8.07E+06	2.14E+07	8.15E+06	1.07E+07	1.76E+07	8.05E+05
u_i_hilo	16499000	33999000	16599000	20499000	27999000	16999000
u_i_lohi	20999000	55299000	21199000	29299000	44599000	25099000
u_i_lolo	662000	1329000	660000	806000	1099000	684000
u_p_hihi	1.12E+07	2.20E+07	3.64E+07	1.56E+06	2.12E+07	1.36E+05
u_p_hilo	23299000	41499000	85699000	28699000	36499000	25199000
u_p_lohi	26699000	62999000	83699000	42199000	56799000	34299000
u_p_lolo	594000	1099000	1949000	839000	907000	657000

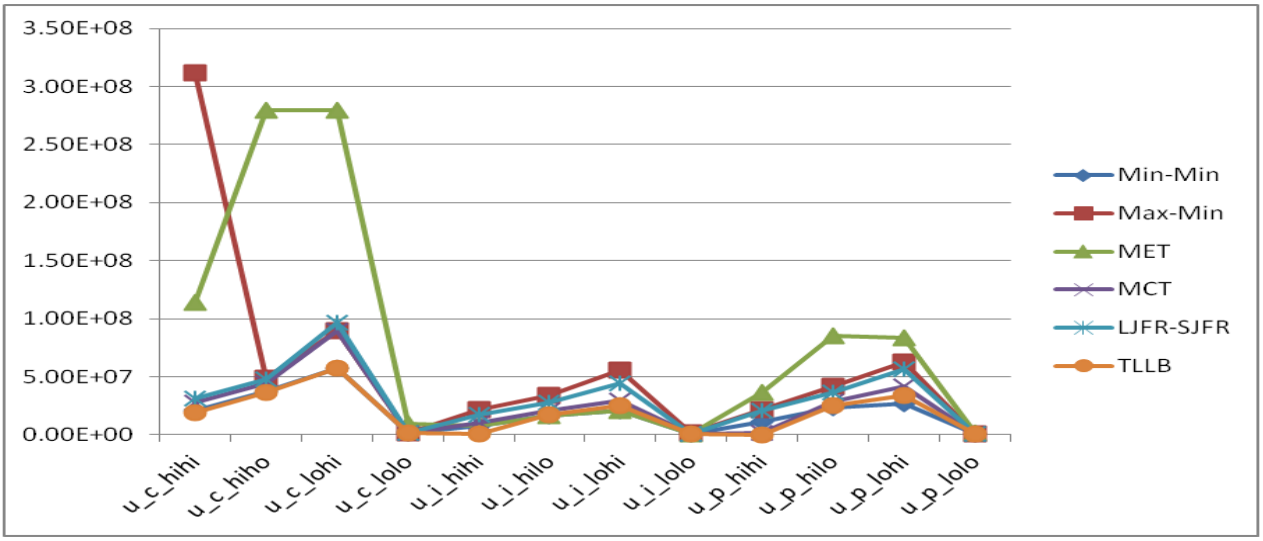

Fig 3 : Graphical Representation of flowtime value

5.3. Resource Utilization

Maximizing the resource utilization of the grid system is another important objective. This criterion is gaining importance due to the economic aspects of grid systems. The algorithm should improve the utilization of resources by reducing the idle time of the resources. One possible definition of this parameter is to consider the average utilization of resources. For instance, in the ETC model, it can be defined as follows:

$$\text{Utilization} = \frac{\sum(i \in \text{resources})^{C_i}}{\text{makespan X nb\_resources}}$$

Table 3 shows the improvement of TLLB algorithm over traditional algorithms. From this figure we can observe that TLLB uses the maximum amount of resources while reducing the make span obtained from Min-Min algorithm. Thus TLLB uses the idle resources to reduce the make span.

Table 3. Resource Utilization in Percentage

Algorithm	Utilization
Min-Min	90.02%
Max-Min	95.04%
MET	89.15%
MCT	90.21%
LJFR-SJFR	94.32%
TLLB	98.52%

6. CONCLUSION AND FUTURE WORKS

To overcome the limitations of Min-Min algorithm, a new task scheduling algorithm TLLB is proposed. It uses the advantages of Min-Min algorithms and covers their disadvantages. The TLLB algorithm and various existing algorithms are tested using the benchmark simulation model for distributed heterogeneous computing systems. The experimental results shows that proposed algorithm TLLB outperforms in makespan, flow time and resource utilization on various heterogeneous environment. In the future, we can extend our scheduling approach by using communication cost between tasks, deadline of tasks, dynamic priority and security mechanisms.

7. REFERENCES

[1] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai, “An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment”, World Academy of Science, Engineering and Technology 29, pp. 314- 321, 2007.

[2] Kokilavani.T and George Amalarethinam.D.I, Applying Non-Traditional Optimization Techniques to Task Scheduling in Grid Computing, International Journal of Research and Reviews in Computer Science, Vol. 1, No. 4, Dec 2010, pp. 34 – 38

[3] He. X, X-He Sun, and Laszewski. G.V, "QoS Guided Minmin Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[4] Sameer Singh Chauhan,R. Joshi. C, QoS Guided Heuristic Algorithms for Grid Task Scheduling, International Journal of Computer Applications (0975 – 8887), pp 24-31, Volume 2, No.9, June 2010.

[5] Dong. F, Luo. J, Gao. L and Ge. L, "A Grid Task Scheduling Algorithm Based on QoS Priority Grouping," In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC’06), IEEE, 2006.

[6] Etminani .K, and Naghibzadeh. M, "A Min-min Max-min Selective Algorithm for Grid Task Scheduling," The Third IEEE/IFIP International Conference on Internet, Uzbekistan, 2007.

[7] Ranganathan, K. and Foster, I., “Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications”, Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC 11),Edinburgh, Scotland, July 2002.

[8] Ullah Munir. E, Li. J, and Shi. Sh, 2007. QoS Sufferage Heuristic for Independent Task Scheduling in Grid. Information Technology Journal, 6 (8): 1166-1170.

[9] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., et al. “A comparison of eleven static heuristics for mapping a class

of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp.810–837, 2001

[10] Maheswaran. M, Ali. Sh, Jay Siegel. H, Hensgen. D, and Freund.R.F, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, Journal of Parallel and Distributed Computing, Vol. 59, pp. 107-131,1999.

[11] Saeed Parsa, Reza Entezari-Maleki RASA: A New Grid Task Scheduling Algorithm , International Journal of Digital Content Technology and its Applications Volume 3, Number 4, December 2009.

[12] F. Xhafa, J. Carretero, L. Barolli and A. Durresi, "Immediate Mode Scheduling in Grid Systems", International Journal of Web and Grid Services, Vol.3 No.2, 219-236, 2007b.

[13] T. Hagerup, "Allocating Independent Tasks to Parallel Processors: An Experimental Study", Journal of Parallel and Distributed Computing, 47, 1997, pp. 185-197.

[14] A. Abraham, R. Buyya, B. Nath, "Nature's heuristics for scheduling jobs on computational grids", The 8th IEEE International Conference on Advanced Computing and Communications, 2000.

[15] F. Xhafa, L. Barolli and A. Durresi, "Batch Mode Schedulers for Grid Systems. International Journal of Web and Grid Services", Vol. 3, No. 1, 19-37, 2007.

[16] Doreen Hephzibah Miriam. D and Easwarakumar. K.S, A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010.

[17] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, and R. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing, vol. 61, no. 6, 2001, pp. 810-837.

[18] F. Xhafa and A. Abraham, "Meta-heuristics for Grid Scheduling Problems," In Meta-heuristics for Scheduling in Distributed Computing Environments, Springer, vol. 146, 2008, pp. 1-37.

[19]