Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing

T. Kokilavani
J.J. College of Engineering & Technology and
Research Scholar, Bharathiar University,
Tamil Nadu, India

Dr. D.I. George Amalarethinam
Reader & Director, Department of MCA
Jamal Mohamed College
Tamil Nadu, India

ABSTRACT
Grid computing has become a real alternative to traditional supercomputing environments for developing parallel applications that harness massive computational resources. However, the complexity incurred in building such parallel Grid-aware applications is higher than the traditional parallel computing environments. It addresses issues such as resource discovery, heterogeneity, fault tolerance and task scheduling. Load balanced task scheduling is very important problem in complex grid environment. So task scheduling which is one of the NP-Complete problems becomes a focus of research scholars in grid computing area. The traditional Min-Min algorithm is a simple algorithm that produces a schedule that minimizes the makespan than the other traditional algorithms in the literature. But it fails to produce a load balanced schedule. In this paper a Load Balanced Min-Min (LBMM) algorithm is proposed that reduces the makespan and increases the resource utilization. The proposed method has two-phases. In the first phase the traditional Min-Min algorithm is executed and in the second phase the tasks are rescheduled to use the unutilized resources effectively.

Keywords
Grid Computing, Load Balancing, Min-Min Algorithm, Meta Task Scheduling.

1. INTRODUCTION
Mixed-machine heterogeneous computing environments [1] are a group of heterogeneous high-performance machines interconnected with high-speed links. They are used to solve a variety of computationally intensive applications that require different computing environments. Computation Grids [2] are considered as the next generation of distributed system. Computation Grids are formed by combining geographically distributed resources and various applications. The users who submit their jobs need not be aware of the location of the resources that are used for executing their jobs.

Currently Grid Computing has evolved as a great potential technology that effectively utilizes the idle time of the resources. Grid is distinguished from traditional distributed computing because of its focus on large-scale resource sharing and high performance orientation. Grid is defined by Ian Foster [3] as flexible, secure, coordinated resource sharing among dynamic collection of individuals, institutions and resources which is referred as virtual organizations.

Most complex scientific, engineering and business problems need huge amount of resources for execution. Grid Computing is considered as the best solution for solving these problems [4]. Grid Computing is also used in application areas like weather prediction, astrophysics, bioinformatics, earth quake research, ground water pollution and multiparticle physics. Since the use of Grid is increased on many fields, many developers and researchers focus on the development of both hardware and software needed for Grid architecture. Some of the challenging issues like scheduling, performance prediction and resource management are important in grid computing area [5]. Scheduling is proved to be one of the NP-hard problems in parallel computing itself. Grid scheduling has its own difficulties because of its nature of heterogeneity in operating systems, architecture, resource providers and resource consumers.

Scheduling [4] 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 [6] 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.

Grid users compose their application as a distributed application. Then the users submit their jobs to Grid Resource Broker. The resource broker then queries the Grid Information Service for the availability of resources and to know their properties. The Grid Resources are registered within one or more information service. The resource broker is responsible for scheduling the jobs on the resources that match job's requirements. After scheduling the resource broker monitors the execution of jobs and after execution it collects the results and send back to the users [6].

Large numbers of task scheduling algorithms are available to minimize the makespan [10], [12], [15], [16], [18], [19]. 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 [1], [10], [15], [16], [17], [19]. These two algorithms work by considering the execution and completion time of each task on the each available grid resource.

The Min-Min algorithm first finds the minimum execution time of all tasks. Then it chooses the task with the least execution time among all the tasks. The algorithm proceeds 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 [7], Max-min algorithm schedules larger tasks first. But in some cases, the makespan may increase due to the execution of larger tasks first. The waiting time of smaller tasks is also increased in Max-min.

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 makespan 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 phase and then reschedules by considering the maximum execution time that is less than the makespan obtained from the first phase.

The remaining parts of this paper are organized as follows: Section 2 presents the related works and several well known scheduling algorithms which are 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 and the prominence of the algorithm is demonstrated through an example. Section 5 compares the scheduling algorithms and presents the results of the comparison. Finally, Section 6 concludes the paper and presents future works.

2. RELATED WORKS

A load balancing algorithm aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. The algorithm tries to distribute the load among all the available resources. At the same time, it aims to minimize the makespan with the effective utilization of resources.

In classical distributed systems comprised of homogeneous and dedicated resources, load balancing algorithms have been intensively studied. But these algorithms will not work well in Grid architecture because of its heterogeneity, scalability and autonomy [8]. This makes load balanced scheduling algorithm for grid computing more difficult and an interesting topic for many researchers.

The Non-traditional algorithms differ from the conventional traditional algorithms in that it produces optimal results in a short period of time. There is no best scheduling algorithm for all grid computing systems. An alternative is to select an appropriate scheduling algorithm to use in a given grid environment because of the characteristics of the tasks, machines and network heterogeneity [6].

Braun et al [1] 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. 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 makespan.

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 makespan 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 machine with the minimum completion time for that particular job is selected. But this algorithm considers the job only one at a time.

Min-Min algorithm starts with a set of all unmapped tasks. The machine 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 makespan.

Max-Min is similar to Min-Min algorithm. The machine 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.

Doreen. D et al., [9] have proposed an efficient Set Pair Analysis (SPA) based task scheduling algorithm named Double Min Min Algorithm which performs scheduling in order to enhance system performance in Hypercubic P2P Grid (HPGRID). The simulation result shows that the SPA based Double Min Min scheduling minimizes the makespan with load balancing and guarantees the high system availability in system performance.

He. X et al., [10] have presented a new algorithm based on the conventional Min-Min algorithm. The proposed algorithm which is called QoS guided Min-Min, schedules tasks requiring high bandwidth before the others. Therefore, if the bandwidth required by different tasks varies highly, the QoS guided Min-Min algorithm provides better results than the Min-Min algorithm. Whenever the bandwidth requirement of all of the

tasks is almost the same, the QoS guided Min-Min algorithm acts similar to the Min-Min algorithm.

Kamalam et al., [11] presents a new scheduling algorithm named Min-mean heuristic scheduling algorithm for static mapping to achieve better performance. The proposed algorithm reschedules the Min-Min produced schedule by considering the mean makespan of all the resources. The algorithm deviates in producing a better schedule than the Min-Min algorithm when the task heterogeneity increases.

Sameer Singh et al., [12] have presented two heuristic algorithms: QoS Guided Weighted Mean Time-Min(QWMTM) and QoS Guided Weighted Mean Time Min-Min Max-Min Selective(QWMTS). Both algorithms are for batch mode independent tasks scheduling. The network bandwidth is taken as QoS parameter.

Singh.M et al., [13] present a QoS based predictive Max-Min, Min-Min Switcher algorithm for scheduling jobs in a grid. The algorithm makes an appropriate selection among the QoS based Max-Min or QoS based Min-Min algorithm on the basis of heuristic applied, before scheduling the next job. The effect on the execution time of grid jobs due to non-dedicated property of resources has also been considered. The algorithm uses the history information about the execution of jobs to predict the performance of non-dedicated resources.

Yagoubi. B et al., [14] have offered a model to demonstrate grid architecture and an algorithm to schedule tasks within grid resources. The algorithm tries to distribute the workload of the grid environment amongst the grid resources, fairly. Although, the mechanism used here and other similar strategies which try to create load balancing within grid resources can improve the throughput of the whole grid environment, the total makespan of the system does not decrease, necessarily.

Among all the algorithms stated the Min-Min algorithm is simple and fast, at the same time it produces a better makespan. But it considers the shortest jobs first so it fails to utilize the resources efficiently which leads to a load imbalance. The aim of this work is to overcome the drawback of the Min-Min algorithm. So a two-phase Min-Min algorithm is proposed which improves the load balancing as well as produces a makespan better than the Min-Min algorithm.

3. PROBLEM DEFINITION

Due to the NP-completeness nature of the mapping problem, the developed approaches try to find acceptable solutions with reasonable cost considering many trade-offs and special cases. In this study, the proposed algorithms have been developed under a set of assumptions:

- The applications to be executed are composed of a collection of indivisible tasks that have no dependency among each other, usually referred to as *metatask.*
- Tasks have no deadlines or priorities associated with them.
- Estimates of expected task execution times on each machine in the HC suite are known. These estimates can be supplied before a task is submitted for execution, or at the time it is submitted.
- The mapping process is to be performed statically in a batch mode fashion.
- The mapper runs on a separate machine and controls the execution of all jobs on all machines in the suite.
- Each machine executes a single task at a time in the order in which the tasks are assigned (First Come First Served - FCFS).
- The size of the meta-tasks and the number of machines in the heterogeneous computing environment is known.

In static heuristics, the accurate estimate of the expected execution time for each task on each machine is known a priori to execution and is contained within an ETC (expected time to compute) matrix where *ETC (ti ,mj)* is the estimated execution time of task *i* on machine *j*.
The main aim of the scheduling algorithm is to minimize the makespan. 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 and

Let Resource set R = m1, m2, m3, …. , mk

Be the set of resources available at the time of task arrival

Makespan produced by any algorithm for a schedule can be calculated as follows:

$$\textbf{makespan} = \max (CT(ti, mj))$$

$$CT_{ij} = R_j + ET_{ij}$$

Where CT → completion time of machines

ET_{ij} → expected execution time of job i on resource j

R_j → ready time or availability time of resource j after completing the previously assigned jobs.

The Load Balanced Min-Min algorithm is developed to work for the above stated problem.

3.1 LBMM

Our proposed grid scheduling algorithm, LBMM, is presented in Figure 1. The algorithm starts by executing the steps in Min-Min strategy first. It first identifies the task having minimum execution time and the resource producing it. Thus the task with minimum execution time is scheduled first in Min-Min. After that it considers the minimum completion time since some resources are scheduled with some tasks. Since Min-Min chooses the smallest tasks first it loads the fast executing resource more which leaves the other resources idle. But it is simple and produces a good makespan compared to other algorithms.

```
for all tasks Ti
 for all resources
    Cij=Eij+rj
 do until all tasks  are mapped
    for each task find the earliest completion time and the
resource that obtains it
    find the task Tk with the minimum earliest completion time
    assign task Tk to the resource Rl that gives the earliest
completion time
   delete task Tk from list
   update ready time of resource Rl
   update Cil for all i
end do
// rescheduling to balance the load
sort the resources in the order of completion time
for all resources R
Compute makespan = max(CT(R))
End for
for all resources
  for all tasks
    find the task Ti that has minimum ET in Rj
    find the MCT of task Ti
    if MCT <  makespan
       Reschedule the task Ti to the resource that produces it
       Update the ready time of both resources
    End if
  End for
End for
//Where MCT represents Maximum Completion Time
```

Figure 1. LBMM Heuristic

So LBMM executes Min-Min in the first round. In the second round it chooses the resources with heavy load and reassigns them to the resources with light load. LBMM identifies the resources with heavy load by choosing the resource with high makespan in the schedule produced by Min-Min. It then considers the tasks assigned in that resource and chooses the task with minimum execution time on that resource. The completion time for that task is calculated for all resources in the current schedule. Then the maximum completion time of that task is compared with the makespan produced by Min-Min. if it is less than makespan then the task is rescheduled in the resource that produces it, and the ready time of both resources are updated. Otherwise the next maximum completion time of that task is selected and the steps are repeated again. The process stops if all resources and all tasks assigned in them have been considered for rescheduling. Thus the possible resources are rescheduled in the resources which are idle or have minimum load.

This makes LBMM to produce a schedule which increases load balancing. Since it compares the maximum completion time with makespan LBMM reduces the overall completion time also. The steps to be carried out in the second phase of LBMM are shown in figure 2.

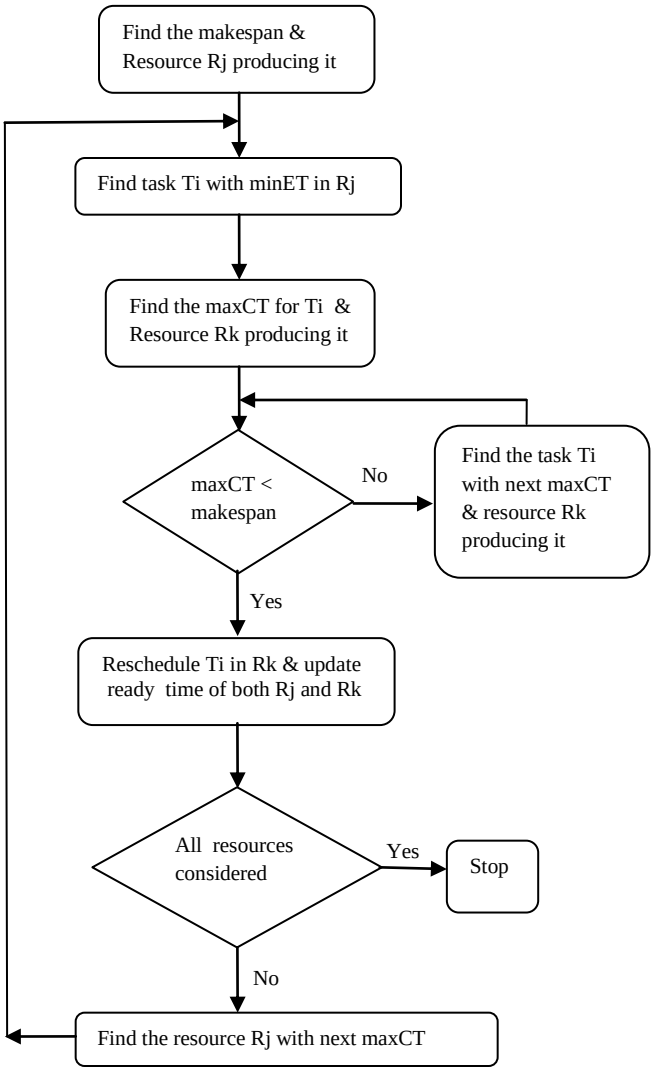

Figure 2. Rescheduling phase of LBMM

4. AN ILLUSTRATIVE EXAMPLE

Consider a grid environment with two resources R1 and R2 and a meta-task group *Mv* with four tasks *T1*, *T2*, *T3* and *T4*. The grid scheduler is supposed to schedule all the tasks within *Mv* on the available resources *R1* and *R2*. Since Min-Min algorithm is simple and produces a better makespan than the other algorithms discussed in the literature, the proposed algorithm executes the Min-Min algorithm in the first phase to schedule the jobs. But to remove the limitation of unbalanced load in Min-Min the jobs are rescheduled in the second phase. In this problem the execution time of all tasks are known prior. They can also be calculated if the number of instructions in each job and the computation rate of each resource is known. They are represented (in sec) in Expected Time to Compute (ETC) table. Table 1 represents the execution time of the tasks on each resource.

Table 1. Expected Execution Time of Tasks

Tasks	Resources	
	R1	R2
T1	7	2
T2	11	3
T3	12	3
T4	6	2

Static mapping of tasks to machines based on Min-Min is shown in Figure 3. Min-Min choose the minimum completion time and so all tasks are scheduled to resource R2 and resource R1 remains idle. The makespan produced by Min-Min is 10 sec.

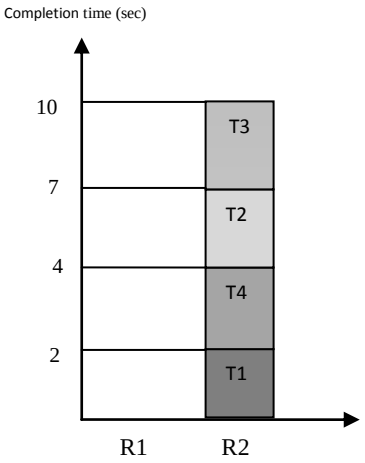

Figure 3. Gantt chart of Min-Min Algorithm

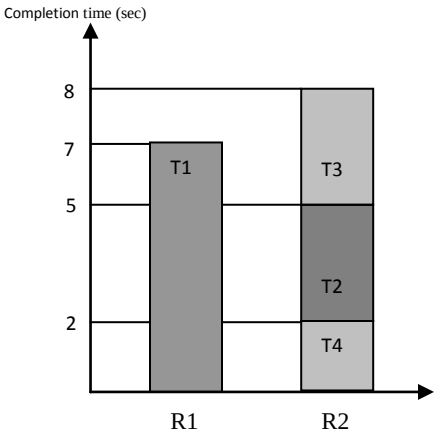

Figure 4. Gantt chart of LBMM Algorithm

According to the proposed LBMM task T1's maximum completion time is less than makespan produced by Min-Min. Other task's maximum completion time is not less than makespan. So task T1 is rescheduled in resource R1 and the remaining tasks are scheduled in the same resource R2. The result of LBMM is shown in Figure 2. Thus the rescheduling of Min-Min algorithm utilizes the idle resource R1 as well as reduces the makespan to 8 sec. Mapping of tasks based on LBMM is shown in figure 4.

5. RESULTS AND DISCUSSION

To evaluate the efficiency of the proposed algorithm, problems having machine heterogeneity and task heterogeneity are collected from various literature [10], [8], [16], [11], [14] and executed for both Min-Min and proposed LBMM algorithm. Interactive software is developed in C++ to execute both algorithms. The results obtained (in sec) for the algorithms are tabulated and shown in Table 2.

Table 2. Comparison of Min-Min and LBMM algorithm

Problem set	Min-Min (sec)	LBMM (sec)
P1	8	6
P2	17	12
P3	33.4	26.6
P4	33.9	32.01
P5	11	10

To show how LBMM outperforms Min-Min the results are plotted in a graph and shown in Figure 5. From this figure we can observe that LBMM produces less makespan than the Min-Min algorithm for all problems.

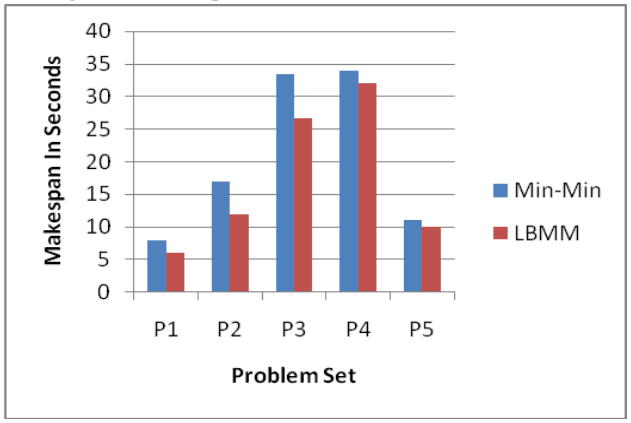

Figure 5. Graphical representation to show improvement of LBMM over Min-Min

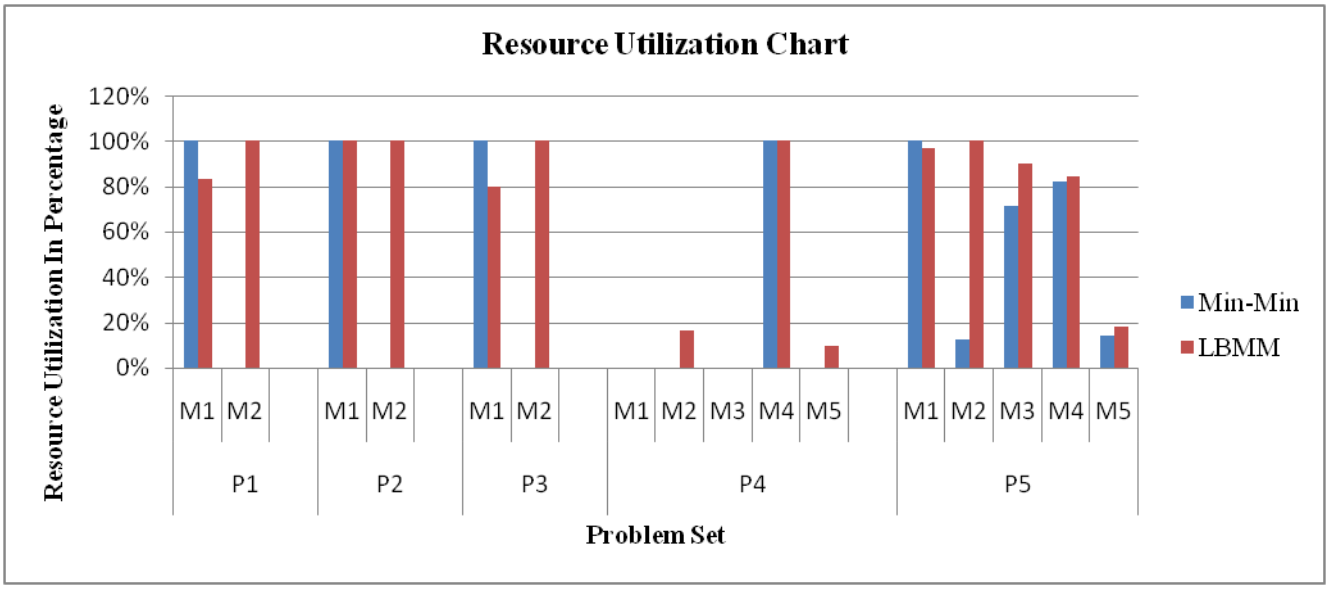

Figure 6. Graphical representation to show more resource utilization of LBMM over Min-Min

Further to show how LBMM balances the load by using the unutilized resource in phase 2 resource utilization of Min-Min and LBMM is calculated for all problems. Table 3 shows the resource utilization rate of both algorithms. From this table we can observe that LBMM tries to use all the available resources. In some problems proposed algorithm uses the same amount of resources, but balances the load in those resources than Min-min. Resource utilization for a particular problem is calculated using the following formula.

$$RU = Mi * 100 / TARU$$
$$TARU = \sum_{i=1}^{n} CTi$$

Here TARU represents Total Amount of Resource Used

The resource utilization rate is represented as graph in Figure 6. From this figure we can observe that LBMM uses the maximum amount of resources while reducing the makespan obtained from Min-Min algorithm. Thus LBMM uses the idle resources for small tasks to reduce the makespan.

6. CONCLUSIONS AND FUTURE WORK

Min-Min and Max-Min algorithms are applicable in small scale distributed systems. When the number of the small tasks is more than the number of the large tasks in a meta-task, the Min-Min algorithm cannot schedule tasks, appropriately, and the makespan of the system gets relatively large. Furthermore it does not provide a load balanced schedule. To overcome the limitations of Min-Min algorithm, a new task scheduling algorithm, is proposed. It is performed in two-phases. It uses the advantages of Max-Min and Min-Min algorithms and covers their disadvantages. The experimental results obtained by applying the proposed algorithm for various problems shows that it outperforms the existing scheduling algorithms. This study is only concerned with the number of the resources and task execution time. The study can be further extended by considering low and high machine heterogeneity and task heterogeneity. Also, applying the proposed algorithm on actual grid environment and considering the cost factor can be other open problem in this area.

Table 3. Resource Utilization in Percentage

Problem Set	Algorithm Used	M1	M2	M3	M4	M5
P1	Min-Min	100	0	-	-	-
	LBMM	83.33	100	-	-	-
P2	Min-Min	100	0	-	-	-
	LBMM	100	100	-	-	-
P3	Min-Min	100	0	-	-	-
	LBMM	80	100	-	-	-
P4	Min-Min	0	0	0	100	0
	LBMM	0	17	0	100	9.52
P5	Min-Min	100	13	72	18	14
	LBMM	97	100	90	85	18.04

7. REFERENCES

[1] 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

[2] Chapman. C, Musolesi. M, Emmerich. W, Mascolo. C, "Predictive Resource Scheduling in Computational Grids" in Parallel and Distributed Processing Symposium, IEEE International Vol. 26, pp.1 – 10, 2007

[3] Ian Foster, Carl Kesselman, Steven Tuecke, "The Anatomy of the Grid Enabling Scalable Virtual Organizations" International Journal of Supercomputer Applications, 2001.

[4] 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.

[5] Dantong Yu and Thomas G. Robertazzi "Divisible Load Scheduling for Grid Computing", PDCS'2003, 15th Int'l Conf. Parallel and Distributed Computing and Systems. IASTED, pp. 1 – 9, 2003.

[6] 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

[7] 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

[8] Geoffrey Falzon, Maozhen Li, "Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments", Journal of Supercomputing, Springer, March 2010.

[9] 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.

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

[11] Kamalam.G.K and Muralibhaskaran.V, , A New Heuristic Approach:Min-Mean Algorithm For Scheduling Meta-Tasks On Heterogenous Computing Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010.

[12] 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.

[13] Singh. M and Suri. P.K, QPS A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid, Information Technology Journal, Year: 2008, Volume: 7, Issue: 8, Page No.: 1176-1181.

[14] Yagoubi. B, and Slimani. Y, "Task Load Balancing Strategy for Grid Computing," Journal of Computer Science, Vol. 3, No. 3, pp. 186-194, 2007.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.