QoS Guided Heuristic Algorithms for Grid Task Scheduling

Sameer Singh Chauhan
Department of Electronics & Computer Engineering
Indian Institute of Technology Roorkee
Roorkee, India - 247667

R. C. Joshi
Department of Electronics & Computer Engineering
Indian Institute of Technology Roorkee
Roorkee, India - 247667

ABSTRACT
Due to the heterogeneity and geographically distribution of Grid resources, effective and efficient task scheduling algorithms are required. Resource load balancing and minimizing makespan are the fundamental goals of effective and efficient task scheduling. It becomes more complicated when various QoS demands arise from users. In this paper, we have presented two algorithms, QoS Guided Weighted Mean Time-min and QoS Guided Weighted Mean Time Min-Min Max-Min Selective, for QoS based Grid task scheduling. Both algorithms consider the resource performance and QoS demands of tasks for scheduling. The algorithms are simulated using GridSim. The results show that the proposed algorithms outperform in makespan, resource utilization and load balancing than other algorithms such as, Weighted Mean Time-min, Weighted Mean Time Min-Min Max-Min Selective, Min-Min, Max-Min and QoS Guided Min-Min.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Distributed Application*

General Terms
Algorithms

Keywords
Grid Computing, QoS, Makespan, Load Balancing.

1. INTRODUCTION
Effective and efficient Task scheduling is an important aspect of Grid computing. Task scheduling becomes more complicated in Grid environment, due to the geographically distribution, heterogeneity and dynamic nature of grid resources. It has been proved that QoS based task scheduling in Heterogeneous Computing (HC) is NP-hard problem[1]. The classical heuristic algorithms such as Min-Min, Max-Min, Sufferage[2], do not consider the QoS demands of tasks in deciding the match between resources and tasks. QoS requirement of a task is one of the important factors in scheduling. QoS guided Min-Min[3] algorithm considers network bandwidth as QoS parameter in scheduling. It divides the tasks in two groups: high and low QoS. It first schedules the tasks from the high and afterword from the low group. The priority grouping algorithm[4], groups the tasks in n groups. These groups are formed on the basis the QoS services provided by the resources. The QoS based algorithms given in [3, 4] shows that the results given by them are better than the classical scheduling algorithms such as given in [2].

In this paper, we 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. The QWMTM divides the tasks in a metatask in two groups: high and low QoS. The heuristic first schedules the high QoS group task and afterward low QoS tasks. The QWMTS heuristic is the extension of our previous work Weighted Mean Time Min-Min Max-Min[5] heuristic. The QWMTS heuristic creates priority groups, to group the tasks with related QoS demands. The groups are created based on the QoS services provided by the resources. If there are n resources, then at most n groups can be created. Each group is assigned a priority level. The tasks are assigned to one of the groups based on their QoS requirements. With the descendent order from high to low priority, the tasks from the group are mapped.

The paper is organized as follows. In section 2 we have discussed the related work. In section 3 the problem is described. In section 4 the QoS Guided Weighted Mean Time-min heuristic is discussed. In section 5 the QoS Guided Weighted Mean Time Min-Min Max-Min Selective heuristic is discussed. In section 6 simulation environment and performance metrics are shown. Results are discussed in section 7. Conclusion and future work is discussed in section 8.

2. RELATED WORK
In this section, we are going to review a set of heuristics those schedules meta-task(MT)[6] to a set of resources. Meta-task can be defined as a collection of independent tasks with no inter-task dependencies. Throughout the paper the number of tasks in MT is represented with n and number of resources is represented with m.

2.1 Min-Min Heuristic
The Min-Min algorithm[2] is a simple algorithm which runs fast and delivers the satisfactory performance. Min-Min begins with the set MT of all unassigned tasks. It 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 resource. Then this task is removed from MT and the process is repeated until all tasks in the MT are mapped. In most situations, it maps as many tasks as possible to their first choice of service resources. However, the Min-Min algorithm is unable to balance the load well since it usually schedules small tasks first. This algorithm takes $O(n^2m)$ time.

2.2 Max-Min Heuristic
Max-Min algorithm[2] is very similar to Min-Min, except in second phase. Max-Min assigns task with maximum expected completion time to the corresponding resource in second phase. The Max-Min algorithm may give a mapping with more balanced

loads across the service resources in some environments. This algorithm takes $O(n^2m)$ time.

2.3 Weighted Mean Time-min Heuristic

Weighted Mean Time algorithm[7] employs weighted mean execution time as heuristic and then assigns the tasks which have maximum weighted mean execution time to the machine with minimum earliest completion time. The heuristics finds the performance of each resource, called the weight of the resource. This weight is used to find the weighted mean time of each task.

2.4 QoS Guided Min-Min Heuristic

QoS Guided Min-Min heuristic[3] is based on the Min-Min heuristic. It considers network bandwidth as QoS parameter. It divides the tasks in two groups: high and low QoS. The idea behind this division is that the tasks requiring high QoS can only run on high QoS providing hosts. The low QoS task can run on any hosts and if they are allocated to high QoS resources, then it leads large makespan, wastage of resources and unbalancing the load. At last, this reduces the overall performance of Grid systems. The QoS guided Min-Min heuristic first schedules the tasks from high QoS group on resources that can provide high QoS as required. Later it schedules the tasks from low QoS group. This algorithm takes $O(n^2m)$ time.

3. PROBLEM DISCRIPTION

Now a day, user's demand for various QoS is continuously increasing in various computing environments. The QoS is an extensive concept and it varies from application to application. It could be the requirement of CPU speed, network bandwidth, deadline, execution cost etc. Providing nontrivial QoS is one of the primary goals of Grid computing. The QoS demands from users put conditions, on the schedulers, to run the applications/tasks. We can justify the QoS demand of a task by following example. Let there are two tasks t1 and t2 and two resources r1 and r2. The task t1 can only be executed on resource r1 but task t2 can be executed on any of the two resources. Now if we schedule task t2 first on resource r1 then task t1 has to wait till task t2 completes. Meanwhile the resource r2 is also idle. Scheduling the tasks in above way increases the makespan, wastes the resources and it eventually results in overall degradation of the performance of Grid. The problem is to design such algorithms which should consider the QoS demands of tasks in scheduling. The algorithm should give preference to tasks with QoS demands and should schedule them first. From the above example, if task t1 is first scheduled on resource r1 and task t2 on resource r2, then the requirement of both tasks is satisfied and the resources are also utilized fully.

Now, we are going to give the terminology [2] used in this paper.

The expected execution time ET_{ij} of task t_i on resource r_j is defined as the amount of time taken by r_j to execute t_i given that r_j has no load when t_i is assigned. The expected execution time matrix (ETC), is formed by finding the expected execution time of each task on every resource. The expected completion time CT_{ij} of task t_i on resource r_j is defined as the wall-clock time at which r_j completes t_i after having finished previously assigned work. The makespan of the schedule is defined as $\max_{ti\in\{t1,\ t2,\ldots,tn\}} CT_{ij}$, where task t_i is assigned to resource r_j. Hence,

$$CT_{ij} = ET_{ij} + rt_j \tag{3.1}$$

Here rt_j is the ready time of resource r_j. Ready time is the time after which the resource will be free to execute a new task.

4. QoS GUIDED WEIGHTED MEAN TIME-MIN HEURISTIC

The Weighted Mean Time-min heuristic, does not consider the QoS demand of tasks for scheduling. The heuristic is modified and a QoS parameter, network bandwidth, is introduced. It considers this QoS parameter as QoS demand of task for scheduling. The modified algorithm is given in figure 4.1.

(1)	Divide the tasks in two groups : High and Low QoS
(2)	**While** there are tasks in MT
(3)	**Do** until all tasks with high QoS in MT are mapped
(4)	**For** each task find the weighted mean time
(5)	Find task t_i with maximum weighted mean time
(6)	For the task t_i, find the resource r_j from QoS qualified resources that gives minimum completion time
(7)	Assign task t_i to resource r_j
(8)	Update ready time of resource r_j
(9)	Delete task t_i from MT
(10)	**End Do**
(11)	**Do** until all tasks with low QoS in MT are mapped
(12)	**For** each task find the weighted mean time
(13)	Find task t_i with maximum weighted mean time
(14)	For the task t_i find the resource r_j that gives the earliest completion time
(15)	Assign task t_i to resource r_j
(16)	Update ready time of resource r_j
(17)	Delete the task t_i from MT
(18)	**End Do**
(19)	**End While**

Figure 4.1 QWMTM Heuristic

The working of the algorithm is as follows: The algorithm first divides the tasks into two groups: high and low QoS. In high QoS group the task with high QoS demands are taken. In low QoS group, tasks with low or no QoS demands are taken. The algorithm first schedules tasks from the high QoS group and afterward tasks from low QoS group. For each group it first calculates the performance of the resources, using equation (4.1), called weight of the resource.

$$w_i = \frac{avg_i}{\sum_{j=1}^{m} avg_j} \tag{4.1}$$

Here avg_i is the average of expected execution time all tasks on each resource r_j.

$$avg_i = \frac{\sum_{i=1}^{n} ET_{ij}}{n} \tag{4.2}$$

The weighted mean time of each task can be calculated using equation (4.3).

$$wmt_i = \sum_{j=1}^{m} w_j ET_{ij} \quad (4.3)$$

For each task in high QoS group, the algorithm calculates the weighted mean time using equation (4.3). It selects the task t_i with maximum weighted mean time for mapping. It finds the resource r_j, from QoS qualified resource set, for the task t_i that gives the earliest completion time. It maps the task t_i on resource r_j. It updates the ready time of resource r_j. Task t_i is deleted from MT. The process continues till all the tasks are mapped. After mapping, all high QoS tasks, the algorithm maps the tasks from low QoS group. For each task in low QoS group the algorithm calculates weighted mean time using equation (4.3). It selects the task t_i with maximum weighted mean time. For the task t_i, it finds the resource r_j that gives earliest completion time. It maps the task t_i on resource r_j. It updates the ready time of resource r_j. Task t_i is deleted from MT. The process continues till all tasks from low QoS group are mapped.

5. QoS GUIDED WEIGHTED MEAN TIME MIN-MIN MAX-MIN SELECTIVE HEURISTIC

The Weighted Mean Time Min-Min Max-Min Selective heuristic[5] considers the performance of resources and calls this performance as weight of resource. It uses the merits and demerits of Min-Min and Max-Min heuristics for scheduling. In its present form it does not consider the influence brought by the QoS of task. We have included a QoS parameter, network bandwidth in it. The modified heuristic QoS guided weighted mean time min-min max-min selective considers the QoS demand of task for mapping. It creates n priority groups of tasks based on the services available at the time of mapping. Each task is assigned to one of the groups based on its QoS demand. Each group is assigned a priority level. Tasks from the highest priority group are mapped first and afterward tasks from high to low priority group are mapped. The algorithm is shown in figure 5.1.

The working of algorithm is as follows. It first creates the expected time compute matrix by calculating the expected execution time of each task on all resources. It computes n groups, based on the QoS services provided by the resources. For each group it calculates the weight of the resources in that group. It calculates the weighted mean time of each task in the group. It calculates the standard deviation of the completion time of unassigned tasks of MT. The standard deviation[8] can be calculated using equation(5.1).

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(CT_{ij} - avgCT)^2}{n}} \quad (5.1)$$

Here avgCT is average of completion time of all unassigned tasks. It can be defined as

$$avgCT = \frac{\sum_{i=1}^{n} CT_{ij}}{n} \quad (5.2)$$

Which task, having maximum or minimum weighted mean time, will be chosen for the mapping that depends on the critical value of the relative standard deviation(SD'). The relative standard deviation can be computed using equation (5.3)

$$SD' = SD/avgCT \quad (5.3)$$

The relative standard deviation shows the degree of dispersion of a set of values, here the set of values are CT_{ij}. If the value of the relative standard deviation is less than the critical value of relative standard deviation(ST), then task with minimum weighted mean time is chosen for mapping otherwise task with maximum weighted mean time is chosen for mapping. The critical value of relative standard deviation can be found by experiments, which come out to be 0.64 in this case.

(1) Get the expected execution time of each task on all resources. Create ETC matrix
(2) Compute n QoS groups
(3) **While** (i < n)
(4) **For** each QoS group
(5) **For** all resources r_j compute

$$avg_j = \frac{\sum_{i=1}^{n} ET_{ij}}{n}$$

(6) **For** all resources r_i compute the weight

$$w_i = \frac{avg_i}{\sum_{j=1}^{m} avg_j}$$

(7) **For** all tasks t_i in group
(8) **For** all resources r_j
(9) $CT_{ij} = ET_{ij} + rt_j$
(10) **For** all tasks t_i, compute the weighted mean time

$$wmt_i = \sum_{j=1}^{m} w_j ET_{ij}$$

(11) Compute the standard deviation (SD) using equation(5.1)
(12) Calculate relative standard deviation SD'
SD' = SD/average(CT_{ij})
(13) **If** SD' < ST then
(14) Find task t_i having minimum weighted mean execution time and assign it to the resource, from the QoS qualified set, that is giving minimum completion time
Else
(15) Find task t_i having maximum weighted mean execution time and assign it to the resource, from the QoS qualified set, that is giving minimum completion time
(16) **Delete** task t_i from the MT
(17) **Update** ready time of resource r_j
(18) **End while**

Figure 5.1 QWMTS Heuristic

6. PERFORMANCE METRICS AND SIMULATION ENVIRONMENT

6.1 Performance Metrics

Depending on what scheduling performance is desired in Grid there exists different performance metrics for evaluating different

scheduling algorithms. Here, the results are evaluated on the basis of following performance matrices.

- **Makespan**: - Makespan is the measure of the throughput of the Grid. It can be calculated using equation (6.1):

$$\text{makespan} = \max_{t_i \in MT}(CT_i) \quad (6.1)$$

The less the makespan, the better is the algorithm.

- **Average resource utilization rate[8] :** Average resource utilization rate of all resources can be computed through equation(6.2)

$$ru = \frac{\sum_{j=1}^{m} ru_j}{m} \quad (6.2)$$

Here the ru_j is the resource utilization rate of resource r_j. It can be computed using equation(6.3)

$$ru_j = \frac{\sum_{i \text{ where } t_i \text{has been executed on } m_j} (te_i - ts_i)}{T} \quad (6.3)$$

Here te_i is the finish time and ts_i is the start time of task t_i on resource r_j. T is the total application time elapsed so far. It can be calculated using equation (6.4)

$$T = \max(te_i) - \min(ts_i) \quad (6.4)$$

- **Load Balancing Level[8] :** The mean square deviation of ru is given by equation (6.5)

$$d = \sqrt{\frac{\sum_{i=1}^{m} (ru - ru_j)^2}{m}} \quad (6.5)$$

The load balancing level, β, is determined through the relative deviation of d over ru.

$$\beta = 1 - \frac{d}{ru} \quad (6.6)$$

The best load balancing level is achieved if β reaches to 1 and d is close to 0.

6.2 Simulation Environment and Data

To evaluate both heuristics, QoS Guided Weighted Mean Time Min and QoS Guided Weighted Mean Time Min-Min Selective, we have used the GridSim Toolkit[9], for simulating the heuristics.

20 resources and batch size of 2000 tasks are taken for each experiment. The arrival of tasks is modeled as Poisson random process. To evaluate both heuristics we have used the following three task scenarios:

Scenario I: - A few short tasks along with many long tasks.
Scenario II: - A few long tasks along with many short tasks.
Scenario III: - Length of tasks is randomly determined.

7. RESULTS

The makespan, average resource utilization rate and load balancing level results of QWMTM and QWMTS heuristics are shown in section 7.1 and 7.2, respectively.

7.1 QWMTM Results

The task scenarios listed in section 6.2 are used for testing the QWMTM heuristic. The results are obtained and compared with the QoS Guided Min-Min(QMinMin), Weighted Mean Time-min(WMTM), Min-Min and Max-Min heuristics.

7.1.1 Makespan Results

Figures 7.1.1(a), 7.1.1(b), 7.1.1(c), show the results for makespan of task scenario I, II and III, respectively. Table 1 shows the comparison of makespan results of QWMTS heuristic with other heuristics. The QWMTM heuristic gives 9.14%, 19.87% and 11.83% better makespan than QMinMin for tasks scenarios I, II, III, respectively. It gives 24.85%, 42.26%, 32.05% gain over makespan than WMTM for task scenario I, II, III, respectively. It gives 26.55%, 50%, 33.79% less makespan than Min-Min for task scenario I, II, III, respectively. It gives 29.83%, 47.6% 34.23% less makespan than Max-Min for task scenario I, II, III, respectively. Overall it gives better makespan than other heuristics.

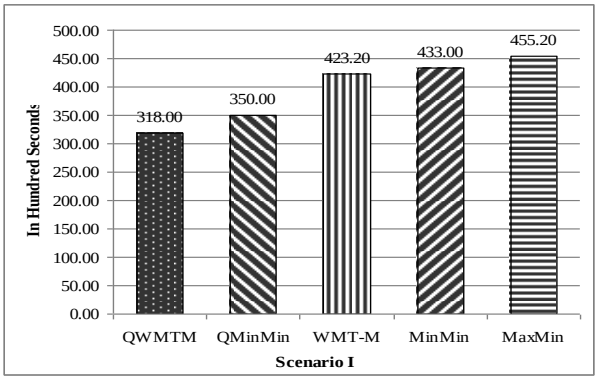

Figure 7.1.1(a) Makespan

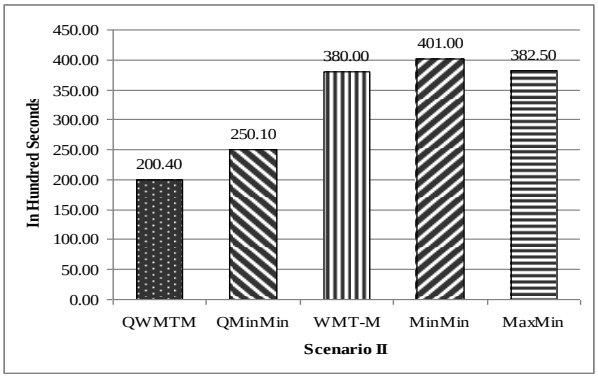

Figure 7.1.1(b) Makespan

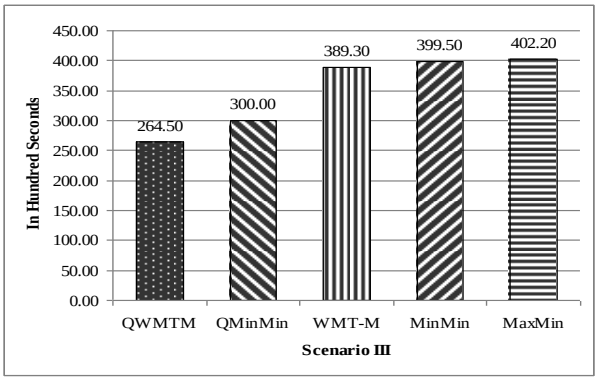

Figure 7.1.1(c) Makespan

Table 1 Makespan Comparison of QWMTM, QMinMin, WMTM, MinMin and MaxMin Heuristics

Task Scenario	Makespan (In Hundred Seconds)					Improvement Over QMinMin	Improvement Over WMTM	Improvement Over MinMin	Improvement Over MaxMin
	QWMTM	QMinMin	WMTM	MinMin	MaxMin				
I	318	350	423.2	433	453.2	9.14%	24.85%	26.55%	29.83%
II	200.4	250.1	380	401	382.5	19.87%	42.26%	50%	47.6%
III	264.5	300	389.3	399.5	402.2	11.83%	32.05%	33.79%	34.23%

7.1.2 Average Resource Utilization Rate Results

Figures 7.1.2(a), 7.1.2(b), 7.1.2(c), showing the results of task scenarios I, II, and III, respectively. We can see the QWMTM gives the better resource utilization results in each task scenario than QMinMin, WMTM, MinMin and MaxMin heuristics.

7.1.3 Load Balancing Level Results

The load balancing results for task scenario I, II and III are shown in figure 7.1.3(a), 7.1.3(b), and 7.1.3(c), respectively. We can see from the results that the proposed heuristic QWMTM provides better load balancing than other heuristics.

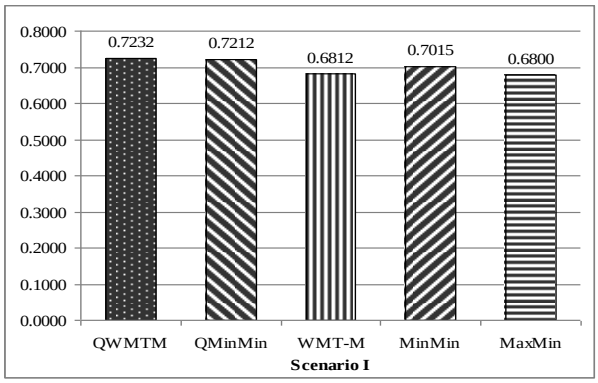

Figure 7.1.2(a) Average Resource Utilization Rate

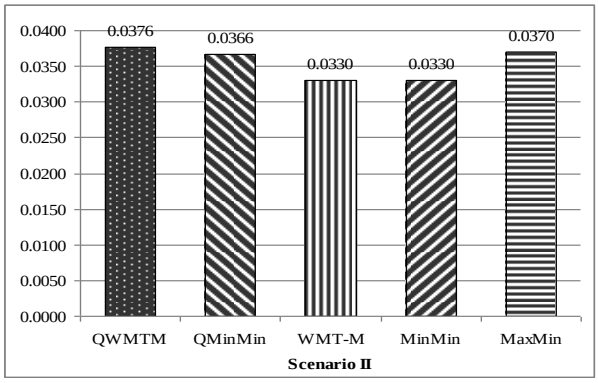

Figure 7.1.2(b) Average Resource Utilization Rate

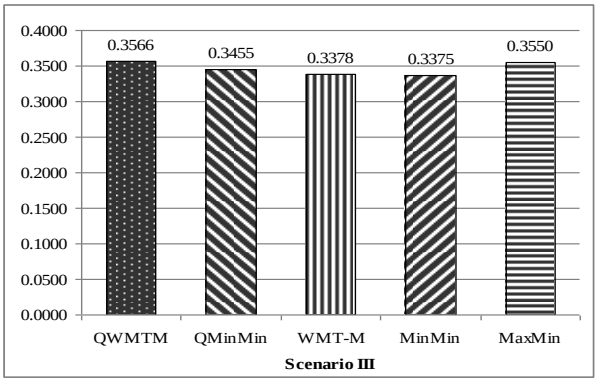

Figure 7.1.2(c) Average Resource Utilization Rate

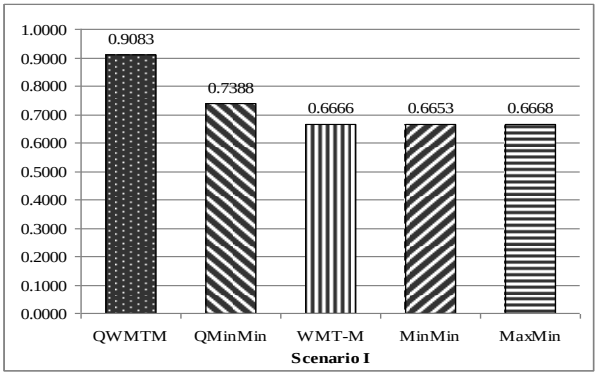

Figure 7.1.3(a) Load Balancing Level

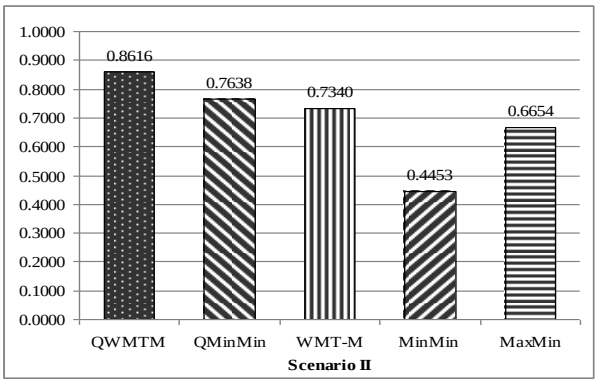

Figure 7.1.3(b) Load Balancing Level

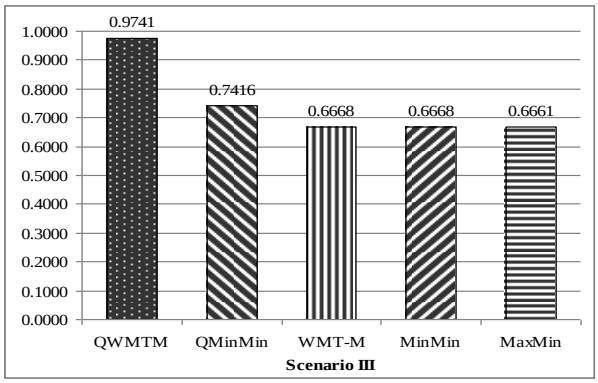

Figure 7.1.3(c) Load Balancing Level

7.2 QWMTS Results

For the testing the heuristic, the task scenarios, given in section 6.2 are taken. The results are compared with the QoS Guided Min-Min(QMinMin), Weighted Mean Time Min-Min Max-Min Selective(WMTS), Min-Min and Max-Min Heuristics.

7.2.1 Makespan Results

Figures 7.2.1(a), 7.2.1(b), 7.2.1(c), show results of makespan for task scenario I, II and III, respectively. Table 2 gives the makespan comparison of QWMTS, QMinMin, WMTS, MinMin and MaxMin heuristics. We can see from the table that QWMTS gives 16.89%, 32.45%, 16.37% shorter makespan than QMinMin heuristic for the task scenario I, II, III, respectively. The QWMTS gives 41.08%, 33.43%, 40.74%, gain over makespan than WMTS for the task scenario I, II, III, respectively. The QWMTS gives 41.08%, 37.37%, 41.42% shorter makespan than MinMin heuristic for the task scenario I, II, III, respectively. The QWMTS gives 41.53%, 33.43%, 41.35% gain over makespan than MaxMin heuristic for the task scenario I, II, III, respectively. The QWMTS heuristic is better in makespan than above mentioned heuristics in every task scenarios.

Table 2 Makespan Comparison of QWMTS, QMinMin, WMTS, MinMin and MaxMin Heuristics

Task Scenario	Makespan (In Hundred Seconds)					Improvement Over QMinMin	Improvement Over WMTS	Improvement Over MinMin	Improvement Over MaxMin
	QWMTS	QMinMin	WMTS	MinMin	MaxMin				
I	263.6	317.2	447.4	447.4	450.9	16.89%	41.08%	41.08%	41.53%
II	146.5	216.9	220.1	233.8	220.1	32.45%	33.43%	37.37%	33.43%
III	264.1	315.8	445.7	450.8	455	16.37%	40.74%	41.42%	41.35%

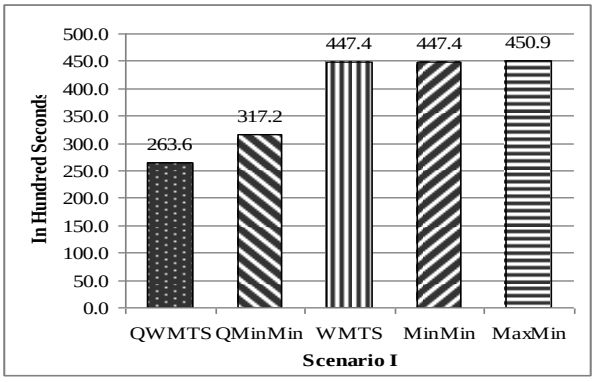

Figure 7.2.1(a) Makespan

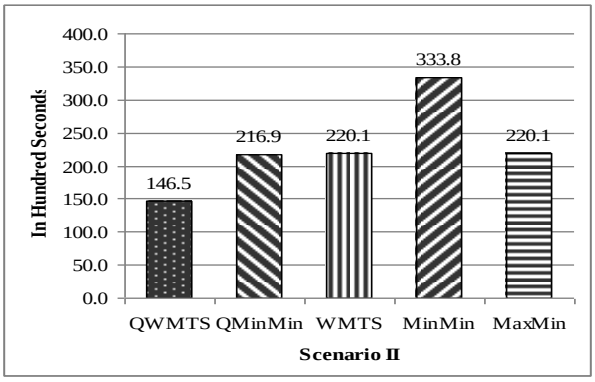

Figure 7.2.1(b) Makespan

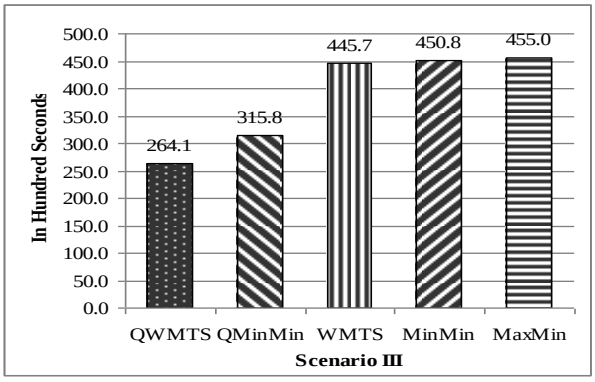

Figure 7.2.1(c) Makespan

7.2.2 Average Resource Utilization Rate Results

The average resource utilization rate results for task scenario I, II, and III are shown in figure 7.2.2(a), 7.2.2(b), and 7.2.2(c), respectively. The results of QWMTS heuristic are better in each task scenario than other heuristics.

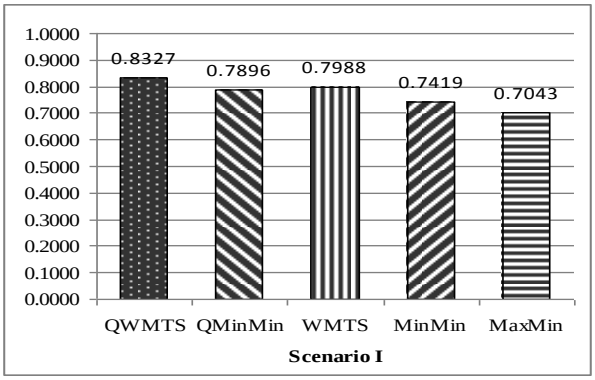

Figure 7.2.2(a) Average Resource Utilization Rate

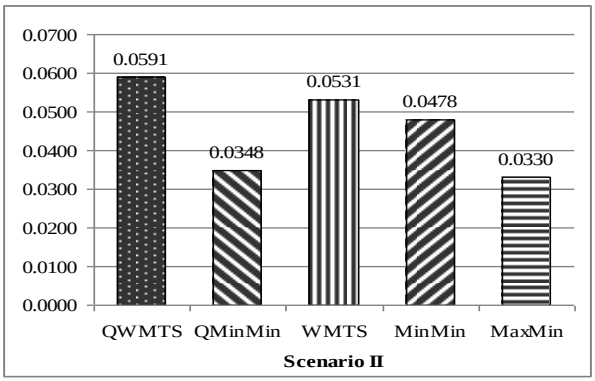

Figure 7.2.2(b) Average Resource Utilization Rate

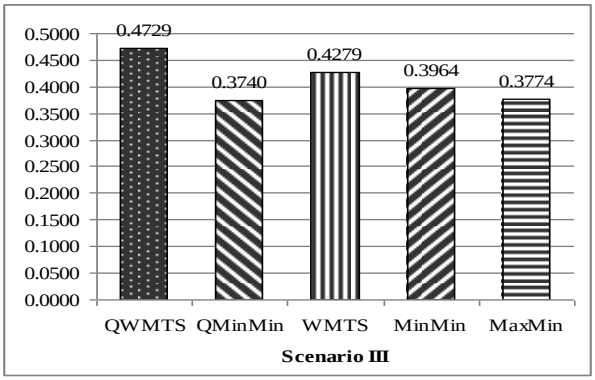

Figure 7.2.2(c) Average Resource Utilization Rate

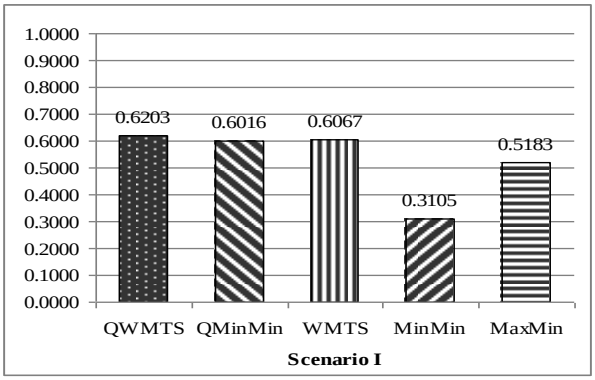

Figure 7.2.3(a) Load Balancing Level

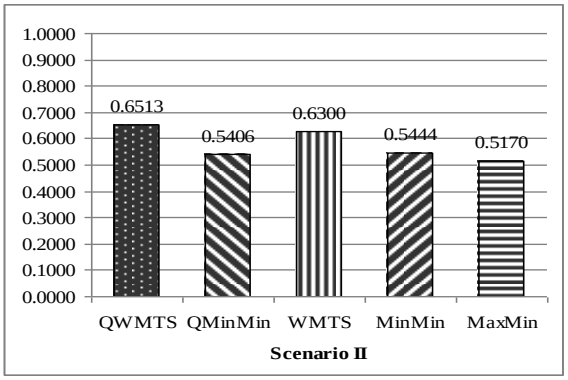

Figure 7.2.3(b) Load Balancing Level

7.2.3 Load Balancing Level Results

The load balancing level results for task scenario I, II, III are shown in figure 7.2.3(a), 7.2.3(b), and 7.2.3(c), respectively. The results of QWMTS heuristic are better than other heuristics in each task scenario.

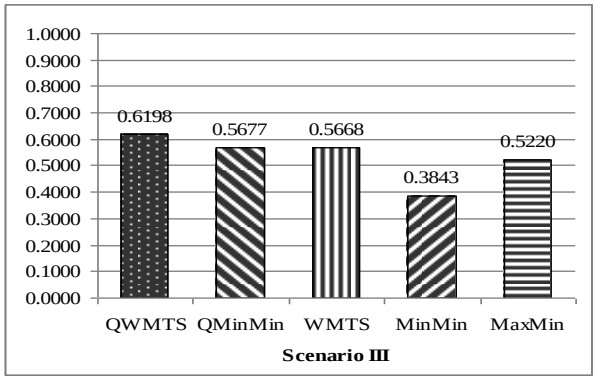

Figure 7.2.3(c) Load Balancing Level

8. CONCLUSION AND FUTURE WORK

In this paper, we have proposed two heuristic algorithms for QoS based task scheduling. QoS Guided Weighted Mean Time-min and QoS Guided Weighted Time Min-Min Max-Min Selective. The QWMTM divides the tasks in two groups: high and low QoS. It schedules the tasks from high QoS group first and afterward tasks from low QoS group. The QoS Guided Weighted Mean Time Min-Min Max-Min Selective heuristic provides the priority grouping strategy to group the tasks with related QoS demand. Table 1 shows the comparison of makespan results of QWMTM with QMinMin, WMTM, Min-Min and Max-Min. The QWMTM gives 9.14% to 19.87% gain over makespan than QMinMin, 24.85% to 42.26% shorter makespan than WMTM, 26.55% to 50% gain in makespan than Min-Min, 29.83% to 47.6% shorter makespan than Max-Min. Table 2 show the comparison of makespan results of QWMTS with QMinMin, WMTS, Min-Min and Max-Min heuristics. QWMTS gives 16.37% to 32.45% gain in makespan than QMinMin, 33.43% to 41.08% shorter makespan than WMTS, 37.37% to 41.42% shorter makespan than Min-Min and 33.43% to 41.53% shorter makespan than Max-Min. Both heuristics provide better makespan, resource utilization and load balancing than above said heuristics.

Adding more QoS parameters in both heuristics is under investigation. Verification of both the heuristics under actual Grid environment can be considered as future problem.

9. REFERENCES

[1] D. Fernández-Baca, Allocating modules to processors in a distributed system, IEEE Transactions on Software Engineering, pp. 1427-1436, November 1989.

[2] Maheswaran M, Ali S, Siegel H J, et al, Dynamic Mapping of a Class of Independent tasks onto Heterogeneous Computing Systems, 8th IEEE Heterogeneous Computing Workshop (HCW '99), Apr. 1999. pp. 30-44.

[3] Xiao-Shan He, Xian-He Sun, QoS Guided Min-Min Heuristic for Grid Task Scheduling, Jouranal of Computer Science & Technology, 2003, (5): 442-451.

[4] Dong, F., J. Luo, L. Gao and L. Ge, A Grid Task Scheduling Algorithm based on QoS Priority Grouping. In proceedings of the 5th International Conference on Grid and Cooperative Computing, 2006, pp. 58-61.

[5] Sameer Singh Chauhan and R. C. Joshi, "Weighted Mean Time Min-Min Max-Min Selective Scheduling Strategy for Independent Tasks on Grid", In proceedings of IEEE 2nd International Advance Computing Conference – 2010 (IACC 2010), pp. 4-9, February, 2010.

[6] Barun TD, Siegel H J and Beck N. 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. 1. PP 810-837, 2001.

[7] Jinquan Z, Lina N, Changjun J, A Heuristic Scheduling Strategy for Independent Tasks on Grid, Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA '05), November 2005.

[8] Kobra Etminani, M. Naghibzadeh. A Min-Min Max-Min Selective Algorihtm for Grid Task Scheduling. Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on, Sept. 2007.

[9] R. Buyya, M. Murshed, GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Journal of Concurrency and Computation: Practice and Experience, pp. 1175–1220, 2002.