Multi Objectives heuristic Algorithm for Grid Computing

Fahd Alharbi
College of Engineering
King Abdulaziz University
Rabigh, KSA

ABSTRACT

Grid computing provides the means of using and sharing heterogeneous resources that are geographically distributed to solve complex scientific or technical problems. Task scheduling is critical to achieving high performance on grid computing environment. The objective of the scheduling process is to map each task with specific requirements to a capable machine in order to minimize the makespan. Task scheduling is shown to be NP-complete problem, which can be solved using heuristic algorithms. Several heuristic algorithms have been proposed in the literature, and they are either not efficient or complex. In this paper, we are proposing a Multi Objectives heuristic Algorithm to minimize the makespan and flow time and to maximize the resource utilization with a low computational complexity.

General Terms
Heuristic algorithms

Keywords
Grid computing, scheduling, makespan, flow time

1. INTRODUCTION

The grid infrastructure provides a mechanism to execute applications over geographically distributed machines by sharing resources, which may belong to different organizations [1-2]. These applications consist of various tasks that must be performed on some capable resources in the grid environment. Grid scheduler receives applications from grid users, selects resources for these applications according to their requirements, and finally maps applications to resources based on certain objective. The grid aims to schedule large number of tasks, with the goal of reducing the tasks' completion time (makespan), balancing the load across the machines [3-6], and maximizing the resource utilization.

The problem of mapping resources to tasks has been shown to be NP-complete [7-10]. These hard combinatorial problems can be solved using the heuristic approach. There are several heuristic algorithms have been proposed to minimize the total completion time of the tasks in grid systems [8, 11]. Among these heuristics, the Min-min algorithm is considered to be simple and achieves the best performance with respect to the makespan. This paper presents the Mact-min for efficient tasks' mapping in the grid computing systems.

The main features of the Mact-min are to achieve minimum makespan, minimum flow time and maximum resource utilization with low computational complexity. The rest of the paper is organized as follows. Section 2 presents the related works. Section 3, the new scheduling algorithm is presented. Section 4 presents some illustrative examples. Section 5 exhibits comparison study among the scheduling algorithms and we conclude at Section 6.

2. Related Works

In this section, we will discuss the scheduling operation in the Grid systems along with the major available heuristic scheduling algorithms and the performance criteria. Consider a Grid system with M tasks to be mapped to N machines. Each machine in the computational grid executes a single task at a time. In static heuristics, the accurate estimate of the expected execution time for each task on each machine is known before scheduling process. The expected time to compute matrix (*ETC*) includes the estimated execution time of task i ($i = 1, 2, ..., M$) on machine j ($j = 1, 2, ..., N$), *ETC (ti, mj)*. When machine *mj* is not capable to execute task *ti*, the value of *ETC (ti, mj)* is set to infinity. The completion time for a task *ti* on machine *mj*, *ct (ti, mj)*, is as follows:

$$ct(ti, mj) = mat(mj) + ETC(ti, mj) \quad (1)$$

where, *mat(mj)* is the machine availability time; the time when machine *mj* complete the execution of all the previously assigned tasks.

2.1 Grid Scheduling Objectives

Now, we discuss several performance criteria to evaluate the quality of the grid scheduling algorithm.

2.1.1 Computational complexity

This criterion will measure how fast the scheduling algorithm in finding the feasible solution in a highly dynamic environment.

2.1.2 Makespan

The main objective of the heuristic scheduling algorithms is to minimize the completion time of last finished task (*makespan*).

The makespan is computed as follows:

$$makespan = max(ct(ti, mj)) \ , i=1,2,\dots,M \ , j=1,2,\dots,N \quad (2)$$

Also, we can compute the makespan as the following:

$$makespan = max(mat(mj)) \ , j=1,2,\dots,N \quad (3)$$

here, *mat(mj)* is the last computed machine availability time; the time when machine *mj* complete execution of all the assigned tasks.

2.1.3 Flow time
Flow time is the sum of all the time needed by all machines to finish all tasks. Flow time is computed as the following:

$$Flowtime = \sum_{j=1}^{N} mat(mj) \qquad (4)$$

The heuristic scheduling algorithm aims to minimize the flow time by minimizing the average task completion time.

2.1.4 Fitness
Minimizing the flow time requires that small tasks are mapped to the fastest machines. Accordingly, the large tasks will take longer completion time, and the makespan will be maximized. On the other hand, minimizing the makespan requires that large tasks are mapped to the quicker machines. Therefore, the small tasks will take longer completion time, and the flow time will be maximized. The fitness criterion will measure the ability of the scheduling algorithm to optimize the makespan ant the flow time. The fitness value is computed as following:

$$Fitness = p * makespan + \frac{(1-p) * flowtime}{N} \qquad (5)$$

where, p is in the range of zero to one based on the importance of the objective. In section 5 we set p to 0.5.

2.1.5 Resource Utilization
Resource utilization is the essential performance criterion for the grid managers. The machine's utilization is defined as the percentage of time that machine *mj* is busy during the scheduling time. The machine's utilization is computed as follows:

$$mu_j = \frac{mat(mj)}{makespan} \quad for\ j = 1, 2, ..., N \qquad (6)$$

The grid's resource utilization is the average of machines' utilization:

$$RU = \frac{\sum_{j=1}^{N} mu_j}{N} \qquad (7)$$

2.2 Grid Heuristic Scheduling Algorithms
Several heuristic algorithms have been proposed to schedule tasks in the Grid computing environment. The commonly used algorithms are discussed in the following.

2.2.1 Minimum Execution Time (MET)
MET (Figure 1) assigns each task in arbitrary order to the machine with the minimum expected execution time for the task regardless of the machine availability time [8]. MET assigns each task its best machine. This leads to severe load unbalance among machines. The heuristic complexity is $O(NM)$, where N and M are the number of machines and the number of tasks respectively.

2.2.2 Opportunistic Load Balancing (OLB)
OLB assigns each task in arbitrary order to the next available machine regardless of the task's expected execution time on the machine [8, 12]. OLB aims to balance the load among the Grid system machines which leads to poor makespan. Furthermore, The OLB heuristic complexity is similar to the MET heuristic.

2.2.3 Minimum Completion Time (MCT)
MCT assigns each task in arbitrary order to the machine with the minimum completion time for the task. Thus, some tasks may not be assigned to the machine with minimum execution time. The assigned task is deleted from the set of tasks, and the completion times for all the remaining tasks are updated. This process continues until all tasks are mapped [8]. Moreover, The MCT heuristic complexity is similar to the MET heuristic.

2.2.4 Min-min
Min-min (Figure 2) first computes the completion time for each task on each machine. Then, the machine with the minimum completion time for each task is selected. Finally, map the task with the minimum completion time to the selected machine. The assigned task is deleted from the set of tasks, and the completion times for all the remaining tasks are updated. This process is repeated until all tasks are mapped [8, 12-14]. Min-min aims to minimize the makespan by assigning tasks with minimum completion time (small tasks) to the faster machines first followed by the tasks with longer completion time (large tasks). This results in a load unbalance and poor utilization. The Min-min heuristic complexity is $O(NM^2)$.

2.2.5 Max-min
Max-min first computes the completion time for each task on each machine. Then, the machine with the minimum completion time for each task is selected. Finally, map the task with the maximum completion time to the selected machine. The assigned task is deleted from the set of tasks, and the completion times for all the remaining tasks are updated. This process continues until all tasks are mapped [8, 13-14]. Max-min achieves better performance better than the Min-min algorithm when the number of small tasks is larger than the number of long tasks. The Max-min heuristic complexity is similar to the Min-min heuristic.

2.2.6 Sufferage
The task's sufferage value is the difference between its best minimum completion time and its second best minimum completion time. The task with high sufferage value is selected and mapped to the machine with the minimum completion time. The assigned task is deleted from the set of tasks, and the completion times for all the remaining tasks are updated. This process is repeated until all tasks are mapped [8, 15]. Since the Sufferage heuristic schedule the task that would highly suffer if it is not assigned to the machine with the minimum completion time first, it is expected to perform well in the systems with machines are highly variant in their execution times for a specific task. Moreover, the sufferage heuristic complexity is similar to the Min-min heuristic.

```
1. for all tasks in the set U
2. for all machines
3. do until all tasks in U are mapped
4.for each task ti in U in an arbitrary order find the machine with the minimum execution time
   Min(ETC (ti, mj)),  $j = 1, 2, ..., N$
5. assign task ti to the machine with minimum completion time
6. delete the selected task from U
7.end do
```

Figure 1: The MET heuristic

```
1. for all tasks in the set U
2. for all machines
3. ct(ti, mj)= mat(mj)+ ETC (ti, mj)
4. do until all tasks in U are mapped
5. find the task ti with the minimum completion time and the machine supports this minimum completion
   min(min(ct(ti,mj))), $i \in unassigned\ tasks$ , $j = 1, 2, ..., N$
6. assign task ti to the machine with minimum completion time
7. delete the selected task from U
8. update mat(mj) for, $j = 1, 2, ..., N$
9.update ct(ti,mj), $i \in unassigned\ tasks$ , $j = 1, 2, ..., N$
10.end do
```

Figure 2: The Min-min heuristic

3. THE PROPOSED HEURISTIC ALGORITHM

In this section, we present the Mact-min heuristic algorithm (Figure 3). Mact-min first computes the completion time for each task on each machine. Then, the task with the maximum average completion time is selected. Finally, map the selected task to the machine with minimum completion time. The assigned task is deleted from the set of tasks, and the completion times and average completion time for all the remaining tasks are updated. This process is repeated until all tasks are mapped. The Mact-min heuristic pseudo code is shown at Figure (3) and the heuristic complexity is $O(NM)$ if the number of machines is larger than the number of tasks, otherwise, the heuristic complexity is $O(M^2)$.

4. ILLUSTRATIVE EXAMPLE

Consider a grid system with three machines and three tasks. The ETC matrix is illustrated in Table 1. The performance of the heuristic algorithms is shown at Figure (4). MET assigns each task to the machine with the minimum expected execution time for the task. Thus, all tasks have been mapped to machine m3 and the makespan is 45. OLB assigns each task in arbitrary order to the next available machine. Accordingly, tasks t1, t2, and t3 have been mapped the machines m1, m2, and m3 respectively and the makespan is 60. MCT assigns each task in arbitrary order to the machine with the minimum completion time for the task. Hence, task t1 is mapped to machine m3, task t2 is assigned to machine m1 and task t3 is assigned to machine m3. Therefore, the makespan is 30. Max-min maps tasks t1 and t3 to machine m3, and task t2 to machine m1. As a result, the makespan is 30. Mapping of the Min-min is illustrated at Figure (5) where the achieved makespan is 30 time units. Similarly, sufferage algorithm achieves a makespan of 30 time units. On the other hand, Mact-min is able to achieve the minimum makespan of 20 time units as shown at Figure (6). Figures (7-8) show that Mact-Min is able to minimize the flow time and to achieve the best resource utilization with a low computational complexity.

```
1. for all tasks in the set U
2. for all machines
3. ct(ti, mj)= mat(mj)+ ETC (ti, mj)
4. for all tasks compute the average completion time
   $act(i) = \frac{\sum_{j=1}^{N} ct(ti, mj)}{N}$   $for\ i = 1, 2, ..., M$
5. do until all tasks in U are mapped
6. find the task with maximum average completion time
      max(act( i)) for $i \in unassigned\ tasks$
7. find the machine mj with the minimum completion time for the selected task ti
      min(ct(ti, mj)) for $j = 1, 2, ..., N$
8. assign task ti to the machine mj
9. delete the selected task from U
10. update mat(mj) for the selected machine mj
11. update ct(ti, mj) for $i \in unassigned\ tasks$ , $j = mj$
12. update act( i)
   $act(i) = act(i) + \frac{ETC(t_i, m_j)}{N}$ , $i \in unassigned\ tasks$
   $ETC(t_i, m_j)$, is execution time of the selected task ti on the selected machine mj
13.end do
```

Figure 3: The Mact-min heuristic

Table 1: The scheduling scenario

	m1	m2	m3
t1	50	20	15
t2	20	60	15
t3	20	50	15

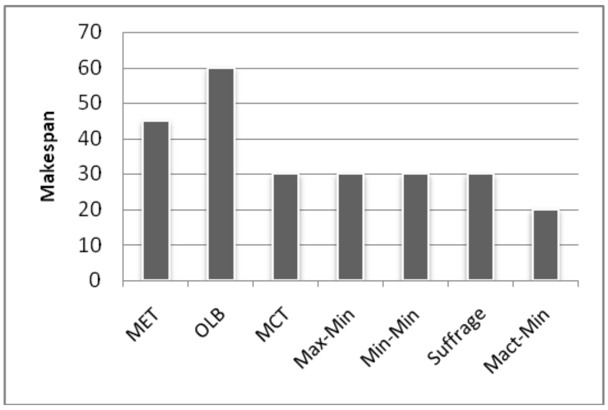

Figure 4: The Makespan

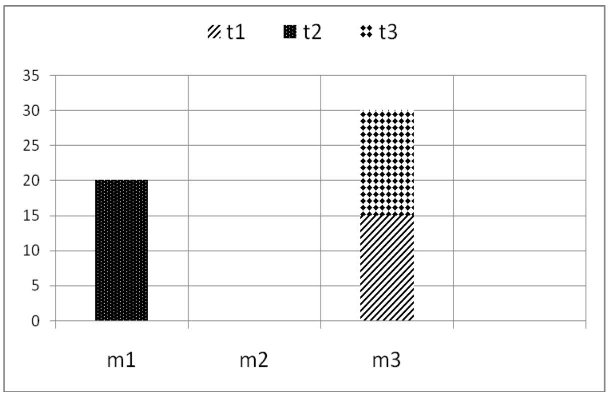

Figure 5: The Min-min heuristic algorithm mapping

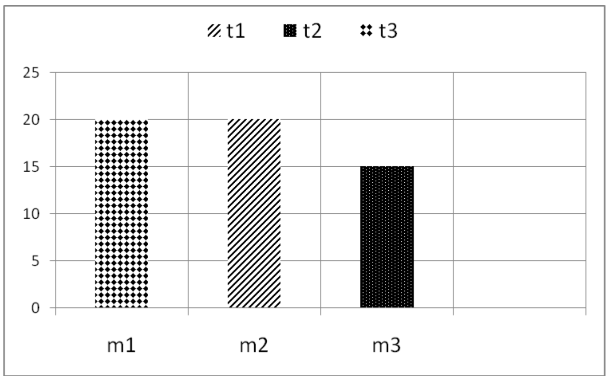

Figure 6: The Mact-Min heuristic algorithm mapping

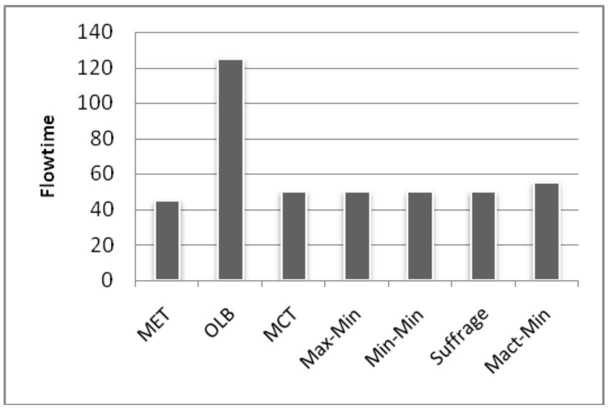

Figure 7: The Flowtime

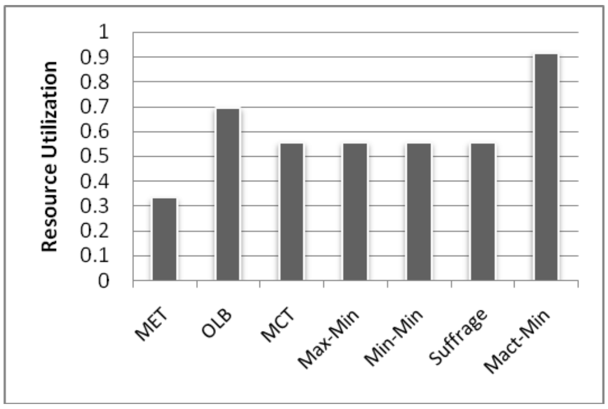

Figure 8: The Resource Utilization

5. PERFORMANCE ANALYSES

For a comparison study among the heuristic algorithms, we are using the ETC model of benchmark simulation experiments [8]. This model is based on Expected Time to Complete (ETC) matrix for 512 tasks and 16 machines. Twelve different instances of the ETC matrices (512x16) are used. These instances are based on task heterogeneity, machine heterogeneity, and consistency. These instances are shown in Table 2. The task heterogeneity indicates the amount of variance among the execution times of tasks for a specific machine. On the other hand, Machine heterogeneity represents the amount of variance among the execution times of machines for a specific task. ETC matrix is consistent if a machine mj executes all tasks either faster or slower than machine mk [16]. In contrast, ETC matrix is inconsistent if machine mj may be faster than machine mk for some tasks and slower for others. Moreover, when some machines are consistent while the other are inconsistent the ETC matrix is semi consistent.

Table 2: the ETC model

Heterogeneity		Consistency		
Task	Machine	Consistent	Inconsistent	semi-consistent
High	High	CHiHi	iHiHi	SHiHi
High	Low	CHiLo	iHiLo	SHiLo
Low	High	CLoHi	iLoHi	SLoHi
Low	Low	CLoLo	iLoLo	SLoLo

The makespan of the heuristic algorithms for the twelve different instances of the ETC matrices are normalized and illustrated at Figures(9-11). The Mact-min achieves performance compatible with the Min-min performance with a lower computational complexity. In contrast, MET and OLB are the worst heuristic algorithms with the respect to the makespan criterion.

Figures (12-14) show the normalized flow times of the heuristic algorithms for the twelve different instances of the ETC matrices. OLB is the worst heuristic algorithms with the respect to the flowtime criterion. On the other hand, Mact-min able to minimize the flow time in most instances.

Figures (15-17) show that Mact-min is able optimize makespan and flow time. In contrast, OLB is the worst in this criterion.

Tables (3-5) show that MET is the worst heuristic algorithms with the respect to the resource utilization criterion. On the contrary, Mact-min achieved the best resource utilization in all instances.

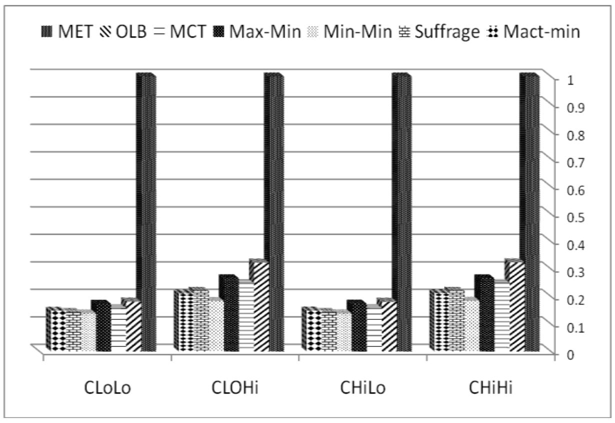

Figure 9: The Makespan (consistent instance)

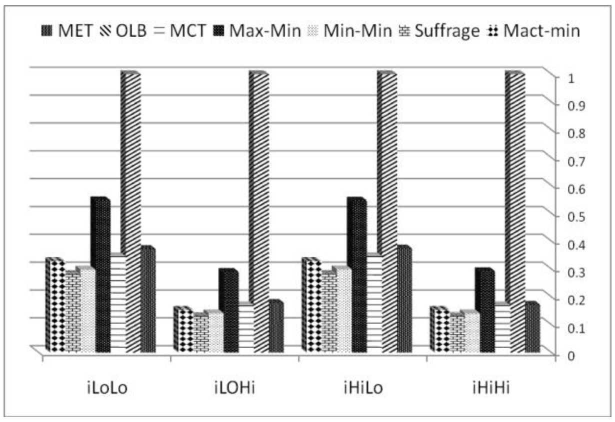

Figure 10: The Makespan (inconsistent instance)

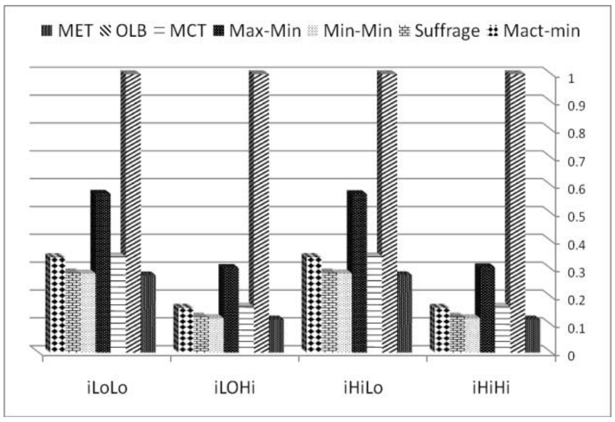

Figure 13: The Flowtime (inconsistent instance)

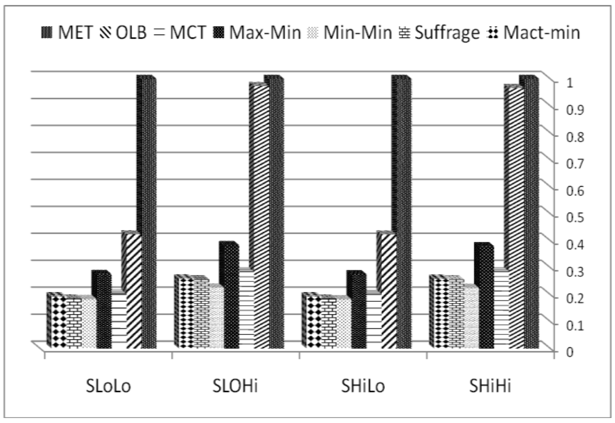

Figure 11: The Makespan (semi-consistent instance)

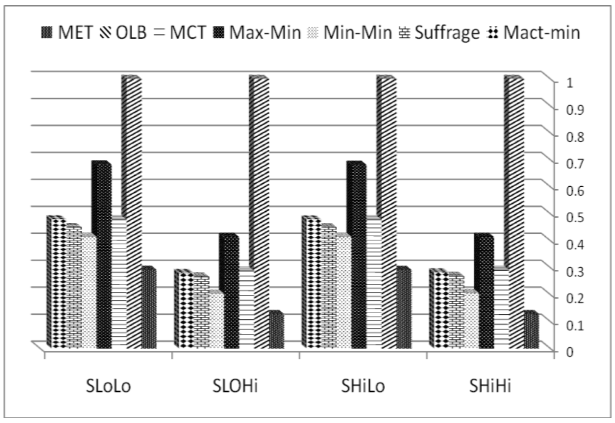

Figure 14: The Flowtime (semi-consistent instance)

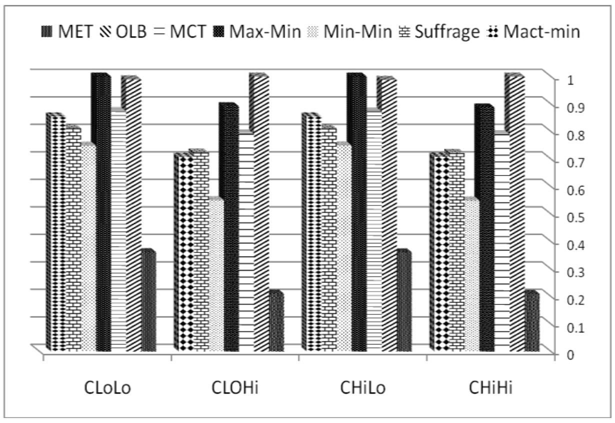

Figure 12: The Flowtime (consistent instance)

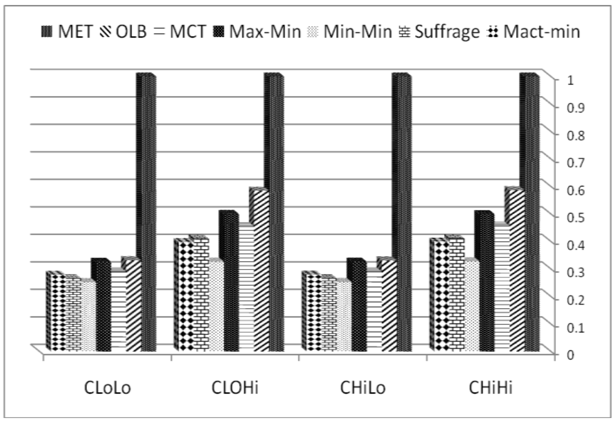

Figure 15: The Fitness (consistent instance)

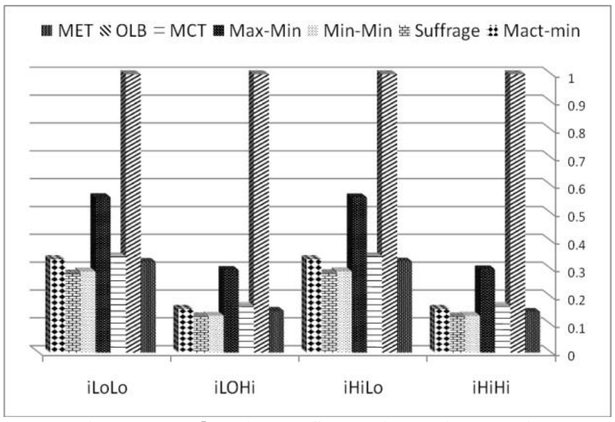

Figure 16: The Fitness (inconsistent instance)

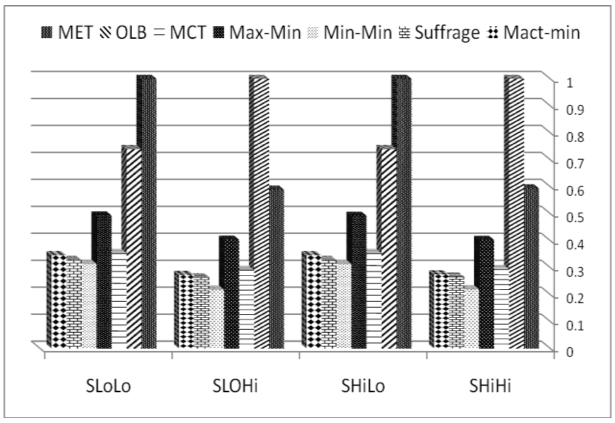

Figure 17: The Fitness (semi-consistent instance)

Table 3: Resource Utilization (consistent instance)

	CLoLo	CLoHi	CHiLo	CHiHi
MET	0.0625	0.0625	0.0625	0.0625
OLB	0.9452	0.9222	0.9472	0.9237
MCT	0.9621	0.9498	0.9617	0.9476
Max-Min	0.9994	0.9992	0.9996	0.9993
Min-Min	0.9364	0.8927	0.9393	0.891
Suffrage	0.9863	0.9761	0.9871	0.9775
Mact-min	0.9994	0.999	0.9995	0.9992

Table 4: Resource Utilization (inconsistent instance)

	iLoLo	iLoHi	iHiLo	iHiHi
MET	0.7201	0.6434	0.7164	0.6673
OLB	0.9581	0.9526	0.9575	0.9524
MCT	0.9561	0.925	0.9565	0.9225
Max-Min	0.999	0.9981	0.999	0.9981
Min-Min	0.9141	0.8338	0.9111	0.8344
Suffrage	0.9767	0.9413	0.9762	0.9351
Mact-min	0.9985	0.9955	0.9987	0.9955

Table 5: Resource Utilization (semi-consistent instance)

	SLoLo	SLoHi	SHiLo	SHiHi
MET	0.1179	0.1183	0.1177	0.1174
OLB	0.9555	0.9512	0.9557	0.9486
MCT	0.9546	0.9357	0.9559	0.9326
Max-Min	0.999	0.9984	0.9991	0.9984
Min-Min	0.9119	0.8334	0.9144	0.8368
Suffrage	0.9812	0.9626	0.9808	0.9609
Mact-min	0.9988	0.9975	0.999	0.9975

6. CONCLUSION

The grid infrastructure provides a mechanism to execute applications over geographically distributed machines by sharing resources, which may belong to different organizations. Task scheduling is critical to achieving high performance on grid computing environment. The scheduling algorithm aims to schedule large number of tasks, with the goal of reducing the tasks' completion time (makespan). Several heuristic algorithms have been discussed. In this paper, we proposed the Mact-min heuristic algorithm to achieve multi objectives, such as minim makespan, minimum flow time, and to maximum resource utilization in the Grid system with a low computational complexity.

7. REFERENCES

[1] I. Foster, and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Second Edition, Elsevier and Morgan Kaufmann Press, 2004.

[2] Zhou Lei and Zhifeng, Allen, Gabrielle Yun, "Grid Resource Allocation," in Grid Computing: Infraestructure, Service, and Applications, Lizhe Wang, Wei Jie, and Jinjun Chen, Eds. Boca Raton: CRC Press, 2009, ch. 7, pp. 1172-188.

[3] Hojjat Baghban, Amir Masoud Rahmani, “ A Heuristic on Job Scheduling in Grid Computing Environment”, In Proceedings of the seventh IEEE International Conference on Grid and Cooperative Computing, pp. 141-146, 2008.3605-7, pp. 8-12, 2009.

[4] Li Wenzheng, Zhang Wenyue, “ An Improved Scheduling Algorithm for Grid Tasks”, International Symposium on Intelligent Ubiquitous Computing and Education, pp. 9-12, 2009.

[5] Hesam Izakian, Ajith Abraham, and Václav Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments," vol. 1, pp. 8-12, 2009.

[6] Fatos Xhafa and Ajith Abraham, Meta-heuristics for Scheduling in Distributed Computing Environments.: Springer-Verlag Berlin Heidelberg, 2008, pp. 1-38, 247-272.

[7] T.Braun, H.Siegel, N.Beck, L.Boloni, M.Maheshwaran, A.Reuther, J.Robertson, M.Theys, B.Yao, D.Hensgen, and R.Freund, “A Comparison Study of Static Mapping Heuristics for a Class of Meta-tasks on Heterogeneous Computing Systems”, In 8th IEEE Heterogeneous Computing Workshop(HCW’99), pp. 15-29, 1999.

[8] Tracy D.Braun, Howard Jay Siegel, and Noah Beck, “A Comparison of Eleven Static Heuristics for Mapping a

Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing 61, pp.810-837, 2001.

[9] Miguel L. Pinedo, Scheduling: Theory, Algorithms, and Systems, Fifth Edition.: Springer, 2008.

[10] Izakian Hesam, Abraham Ajith and Snasel Vaclav, Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments, The 2009 IEEE International Workshop on HPC and Grid Applications (IWHGA2009), China, IEEE Press, USA, ISBN 978-0-7695- 3605-7, pp. 8-12, 2009.

[11] J.M.Schopf, "A General Architecture for Scheduling on the Grid", special issue of JPDC on Grid Computing, 2002.

[12] R.F.Freund, and M.Gherrity, "Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net", In Proceedings of the 7th IEEE HCW, 1998.

[13] R.Armstrong, D.Hensgen, and T.Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions", In 7th IEEE Heterogeneous Computing Workshop(HCW'98), pp. 79-87, 1998.

[14] R.F.Freund and H.J.Siegel,"Heterogeneous Processing", IEEE Computer, 26(6), pp. 13-17, 1993.

[15] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, J. Parallel Distributed Computing 59, 2 (Nov. 1999), 107_121

[16] J.Brevik, D.Nurmi, and R.Wolski, "Automatic Methods for Predicting Machine Availability in Desktop Grid and Peer-to-Peer Systems", In Proceedings of CCGRID'04, pp. 190-199, 2004.