J. Basic. Appl. Sci. Res., **3(4)173-181, 2013**
© 2013, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Batch Mode Scheduling Algorithm for Grid Computing

Salman Meraji*, M. Reza Salehnamadi

Department of Computer Engineering, Islamic Azad University, Tehran South Branch

ABSTRACT

Nowadays, computational grids have been wildly used to share geographically distributed heterogeneous resources, because of high demand of computational power. One of the most important problems in computational grids is scheduling. Max-min and min-min are simple and well known scheduling algorithms for grid computing. The drawback of min-min algorithm is the schedule produced by min-min is not optimal with respect of load balancing and max-min's relative time to finish assigning tasks is too high. To overcome these drawbacks, a new two phase grid scheduler is proposed with simple min-min algorithm in the first phase and a new rescheduling technique in the second phase. To evaluate the performance of proposed algorithm and comparing it with other algorithms, Three main objective functions are measured in grid scheduling which are make span, resource utilization and matching proximity. Algorithms are tested using Expected to compute model of benchmark. The results of simulation shows improvement in all three objective functions i.e. make span is minimized while resource utilization and matching proximity are maximized.

KEYWORDS: Scheduling; Computational Grids; min-min; max-min; make span; resource utilization; matching proximity.

1. INTRODUCTION

The computational grid has provided a reliable infrastructure that helps researchers solving massive applications over geographically distributed nodes by sharing heterogeneous resources such as super computers, work stations, networks, softwares and even simple desktop workstation [1]. Scheduling is one of the most important tasks of computational grid. While Sahu *et al.*[2] introduced five major objective functions and metrics to assess a grid scheduler performance, resource utilization, makespan and matching proximity looks usually have been used for performance measurement. Resource utilization is measured based on idle time of resources and effective job scheduling is measured by high resource throughput. Moreover, makespan is the time difference between the start time of the first job and the finish time of the last job [3], [4]. Matching proximity indicates the degree of proximity of a given schedule to the schedule produced by the Minimum Execution Time (MET) method [2]. The main goal of this research is to design and implement a new scheduling algorithm to minimize makespan and maximizing resource utilization and matching proximity.

Braun *et al.*[5], [6] have studied the relative performance of eleven static heuristic but they have just considered makespan as comparison criterion. Izakian *et al.*[7] have provided a comparison of six heuristic including min-min and max-min with respect of makespan and flow time criteria. They also proposed a new heuristic [8] later named min-max. Xhafa *et al.* compared two types of scheduling algorithms in immediate mode aka online mode [9] and batch mode [10]. Their two researches concerns more than two scheduling objective functions in grid computing. Sahu *et al.* [2] provides a complete analogy of twelve different scheduling algorithm and introduced five major objective function to evaluate them. finally Alharbi [11] proposed a new scheduling algorithm named mact-min and compared it with min-min and max-min. Scheduling of tasks on heterogeneous grid resources is an NP-complete problem, so choosing best heuristic can be applied with respect of MT(Meta Tasks) conditions and available resources. The applications to be executed are composed of a collection of indivisible tasks that have no dependency among each other, usually referred to as MT. To increase efficient usage of resources and mapping tasks different scheduling algorithms have been proposed. Among these heuristics, min-min [5], [6], [7], [8], [10], [11], [12] and Genetic algorithm [5], [6] achieve the best results in makespan and max-min for resource utilization [10], [13].

The main goal of this study is to propose a new scheduling algorithm to improve task assignment performance by schedulers through minimizing makespan amounts and maximazing resource utilization percentage and matching proximity. Generally, Grid scheduling algorithms can be divide in two major groups : Online Mode and Batch Mode [8]. In online mode, tasks are mapped to the first available resources serially. MET(Minimum Execution Time) and MCT(Minimum Completion Time) are the most popular algorithms of online mode scheduling. In batch mode, tasks are grouped together in MT and then the group is scheduled in some predefined times called mapping events. Many heuristics have been proposed for batch mode scheduling among which min-min and max-min are the simplest and most popular ones [14]. Unlike traditional scheduling algorithms which make steady decisions in order to assign a single task to a resource, the propose algorithm using state of the art rescheduling technique. One of the biggest difficulties in grid computing is dynamic nature of grid resources. Rescheduling is a technique to relieve the dynamic problem. The propose scheduler decision

***Corresponding Author:** Salman Meraji Department of Computer Engineering, Islamic Azad University, Tehran South Branch Tel: +98_912_271_1510 Fax : +98_21_220_89327 E-Mail: salman_meraji @yahoo.com

making process to assign tasks is update in each step according to new MT conditions. The drawback of min-min algorithm is the schedule produced by min-min is not optimal when the number of smaller tasks are more than the large tasks. That's because of min-min chooses smaller tasks first which makes use of resource with high processing power and makes resources load imbalance. To overcome this limitation, a two step scheduling algorithm propose. In first step the proposed algorithm run min-min algorithm and then in the second phase reschedule tasks with respect to the makespan and the resource that produced it. To evaluate our proposed algorithm it is compared with min-min that obtained one of the best in makespan and matching proximity and max-min in terms of resource utilization. The proposed algorithm outperforms min-min and max-min in all of these three performance criteria.

The scientific contribution of this study are as follows. 1. proposing a new scheduling schema based on rescheduling technique. 2. Considering multi conflicting objectives to evaluation and capture different aspects of scheduling problem in grid computing. 3. Improving the overall makespan amounts by 13% compared to min-min. 4. Improving the overall resource utilization percentage compared to max-min. 5. Using matching proximity formula to calculate the degree of proximity of a given schedule to the schedule produced by the MET method.

The rest of the paper is organized as follows. In section 2, Grid scheduling algorithms will discuss briefly. Section 3 studies the main objectives of Grid Scheduling. In section 4 the new scheduling algorithm is presented and also some illustrative examples are discussed. Section 5 shows computational results and a comprehensive comparison between our proposed algorithm and two well known algorithms, min-min and max-min.

2- Grid Scheduling algorithms

In this section some well known heuristics for grid scheduling are explained briefly.

2-1 MET(Minimum Execution Time)

MET assigns each task, in an arbitrary order, to the machine with the best expected execution time for that task. In this algorithm a task is assigned to the machine on which it can be executed in minimum time regardless of that machine's availability. Allocating job without considering machine availability might lead to load imbalance on grid machines [2], [5], [6].

2-2 MCT(Minimum Completion Time)

MCT assigns each task, in arbitrary order, to the machine with the minimum expected completion time (ready time of machine + job execution time on the selected machine) for that task. Allocating job in this manner may result in execution of jobs on less faster grid machines [2], [5], [6].

2-3 Min-Min

Min_min heuristic begins with the set U of all unmapped tasks. Then, the set of minimum completion times, M, is found. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine. Finally, the newly mapped task is removed from U, and the process repeats until all tasks are mapped (i.e., U is empty) [5], [15]. The main difference between min-min and MCT is that min-min considers all the unmapped tasks during mapping decision but MCT only considers one task at a time. min-min procedure is shown in algorithm 1. Min-min flowchart is shown in figure1

```
1. For all tasks $t_i$ in $MT$ (in an arbitrary order)
2.     For all Machines $m_j$ (in a fixed arbitrary order)
3.         $C_{ij} = E_{ij} + r_j$
4. Do until all tasks in $MT$ are mapped
5.     For each task in $MT$ find the earliest completion time and the machine that obtains it.
6.     find the task $t_k$ with the minimum earliest completion time.
7.     assign task $t_k$ to the machine $m_l$ that gives the earliest completion time.
8.     delete task $t_k$ from $MT$
9.     update $r_l$
10.    update $C_{il}$ for all $i$
11.End Do.
```

Algorithm 1. Min-Min Algorithm

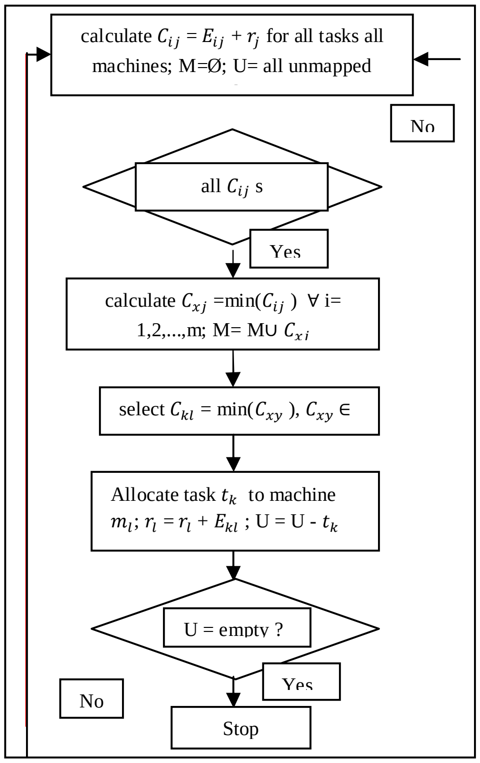
calculate $C_{ij} = E_{ij} + r_j$ for all tasks all machines; M=Ø; U= all unmapped
No
all C_{ij} s
Yes
calculate C_{xj} =min(C_{ij}) ∀ i= 1,2,...,m; M= M∪ C_{xi}
select C_{kl} = min(C_{xy}), C_{xy} ∈
Allocate task t_k to machine m_l; $r_l = r_l + E_{kl}$; U = U - t_k
U = empty ?
Yes
No
Stop

Fig. 1 The Min-Min work flow

2-4 Max-Min

Max-min is very similar to min-min. The Only difference between them is that in step 2, max-min selects the task with overall maximum completion time instead of minimum completion time. The pseudo code for the algorithm is like algorithm 1 just instead of minimum in line 6 is replaced by maximum [5], [15].

3- Grid Scheduling Objectives

Due to the NP-completeness nature of the mapping problem, the proposed heuristics try to find acceptable solutions that tradeoff between cost and performance. Static heuristic algorithms have been developed under some assumptions. The following assumptions are gathered from various literatures [5], [13]:

- Each resource executes only one task at a time i.e. there is no multi-tasking between resources.
- The number of tasks and resources is known prior.
- Estimated of tasks expected execution times are known before or at the time the task is submitted.
- 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.
- There are no priorities or deadlines between the tasks.
- The estimated times store in separate machine. The scheduler also runs on a separate machine and controls the execution of jobs

3-1 Makespan

In static heuristics, the accurate estimate of the expected execution time for each task on each machine is known in advance. These times are stored in an ETC (Expected Time to Compute) matrix where ETC (t_i , m_j) is the estimated execution time of task *i* on machine *j*. The main objective of the heuristic scheduling algorithms is to minimize the completion time of last finished task.
The makespan is computed as follows:
Let task set T = $t_1, t_2, t_3, \ldots, t_n$
be the group of tasks submitted to scheduler and
Let Resource set R = $m_1, m_2, m_3, \ldots, m_k$
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:

(1) makespan = max(CT (t_i , m_j))
(2) $CT_{ij} = R_{j+}ET_{ij}$

where CT_{ij} is Completion Time of task i on resource j, ET_{ij} expected execution time of job i on resource j. and R_j is the ready time or availability time of resource j; the time when machine *mj* complete execution of all the prior assigned tasks.

3-2 Matching Proximity

Matching Proximity is one of the grid performance parameters. Matching proximity indicates the degree of proximity of a given schedule to the schedule produced by the MET method which assigns a job to the machine having the smallest execution time for that job. Matching proximity is an additional performance parameter of batch mode methods. A large value for matching proximity means that a large number of jobs is assigned to the machine that executes them faster [2]. It define as follows:

(3) Matching Proximity= $\dfrac{\sum_{i \in Tasks} ETC[i][Schedule\,][i]}{\sum_{i \in Tasks} ETC[i][MET[i]]}$

3-3 Resource Utilization

Resource utilization is very essential criterion for the users and grid managers. The resource utilization is defined using the completion time of a machine, which indicates the time at which machine m will finalize the processing of the previous assigned jobs as well as those already planned for the machine [2]. It is defined as follows:

(4) Resource Utilization= $\dfrac{\sum_{i \in Machines} \text{Completion[i]}}{\text{makespan.nb}-\text{machines}}$

In formula (4), i indicates the machines, Completion[i] is the completion time of the final job on machine i. Nb-machines indicates the number of machines and makespan is calculated using formula (1)

4- The Proposed Algorithm

In this section, the proposed scheduling algorithm, is called Best-Min. The best-min algorithm uses min-min to get the makespan in first step and reschedule the tasks in the second phase in order to reduce the makespan. There is a condition in algorithm that best-min should consider all the resources in grid environment and this is caused to maximize resource utilization as well.

4-1 best-min algorithm

In this section, the proposed scheduling algorithm will be described. The proposed algorithm uses the state of the art rescheduling technique. The algorithm is presented in algorithm 2. At the first step, best-min, starts with executing min-min. As described in section 2-3, since min-min starts mapping the tasks with minimum execution times first, it makes faster resources busy and slower resource idle. As a result, min-min produces a good makespan comparing to other heuristics. On the other hand, the main drawback of the min-min algorithm is its poor load balanced and resource utilization. Hence, best-min runs min-min first to find the makespan of the whole system and the resource (R_j) that produces that makespan. Makespan and R_j are used in the rest of the best-min algorithm. In the second step, the best-min finds the tasks that are assigned to R_j according to ETC matrix or MT conditions, and chooses the task(T_i) with maximum execution time(Max ET) on R_j. Then, the completion time of the task T_i is calculated for all the other resources.

The minimum completion time of T_i on all the resources is called Min CT. Min CT is compared against the makespan produced by min-min. If Min CT is less than makespan then the task is rescheduled on the resource that produced it and the available time for all resources is updated. Otherwise, T_i is scheduled in current resource(R_j) and scheduler looks for the next task with Max ET. This will be continued till all resources have been considered and all tasks have been mapped.

Comparing the Min CT of the task T_i on other resources and the makespan produced by min-min results in best-min's makespan to never be bigger than min-min's makespan. Also best-min has a good resource utilization. As mentioned, the reason for this is that the scheduler should consider all the available resources to schedule tasks.

The best-min flowchart is also shown in figure 2. Figure 2 only shows the second phase of proposed algorithm since best-min runs min-min in the first step.

```
1. For all tasks ti in MT   (in an arbitary order)
2.     For all Machines mj (in a fixed arbitrary order)
3.         Cij = Eij + rj
4. Do until all tasks in MT are mapped
5.     For each task in MT find the earliest completion time and the machine that obtains it.
6.         Find the task tk with the minimum earliest completion time.
7.         Assign task tk to the machine ml that gives the earliest completion time.
8.     Delete task tk from MT
9.     Update rl
10.    Update Cil for all i
11.End Do.
12.For all Machines
13.Compute Makespan = max(CT(R))
14.End For // The machine that produced the makespan is identified as Rj
15.For all Machines
16.    For all Tasks
17.        Find the task Ti with the Max ET in Rj
18.        Find the Min CT of task Ti
19.    End for
20.    If Min CT < Makespan
21.        Update the Ready Time of both machines.
22.        Reschedule task Ti to the Machine that produces it.
23.    End if
24.End for.
```

Algorithm 2. Best-Min Heuristic

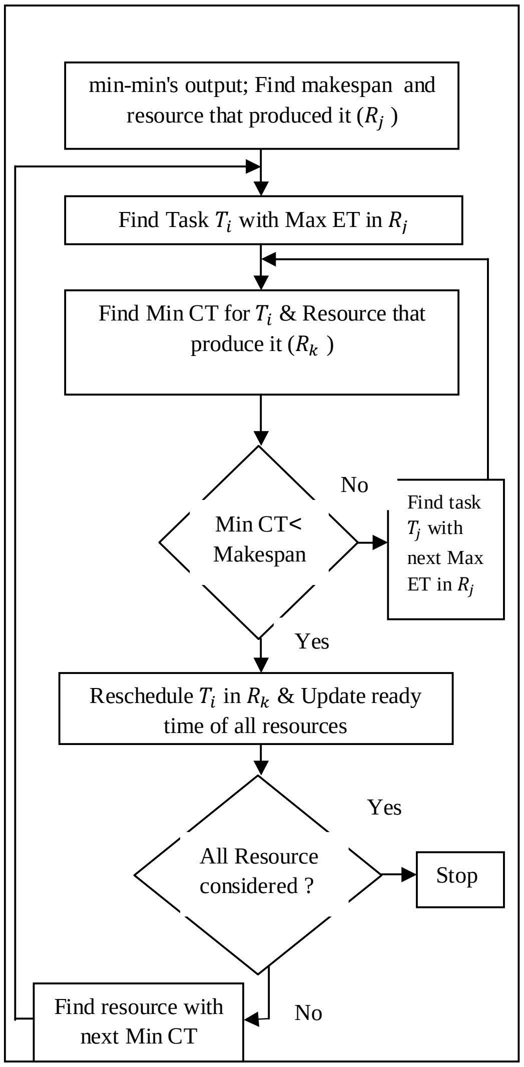
min-min's output; Find makespan and resource that produced it (R_j)
Find Task T_i with Max ET in R_j
Find Min CT for T_i & Resource that produce it (R_k)
Min CT< Makespan
No
Find task T_j with next Max ET in R_j
Yes
Reschedule T_i in R_k & Update ready time of all resources
All Resource considered ?
Yes
Stop
No
Find resource with next Min CT

Fig. 2 The Best-min work flow

4-2 An Illustrative Example

In this section, best-min algorithm is checked by an example. Consider a grid environment with three resources and three tasks. The ETC matrix for this grid system is defined in Table 1. The performance of the different heuristic algorithms is shown at figure 3. MET assigns each task to the resource with the minimum execution time. So all the tasks are assigned to R_3 and makespan becomes 67. Max-min assigns T_1 and T_3 to R_3 and T_2 to R_1 with a makespan of 45. Min-min assigns T_1 and T_2 to R_3 at first and finally maps T_3 to R_3 therefore the makespan is 44. Min-min algorithm is illustrated at figure 4.

Best-min algorithm assigns T_3 to R_1 in the first step. In the second step T_1 is mapped to R_2 and finally T_2 is assigned to R_1 and makespan is reduced to 25. In the first phase according to min-min algorithm makespan is 44 time units and R_3 is the resource that produce it. In the second phase the task with maximum execution time (Max ET) is identified and that is T_3. The Min CT of T_3 is 23 and on R_1 with respect of all the resources except R_3. So T_3 is rescheduled to R_1. The second Max ET in R_3 is T_2. Min CT of T_2 in R_1 and R_2 is 70. Comparing the Min CTs of T_2 (70) in R_1 against R_2 with makespan (44) prove that T_2 has to schedule in current resource (R_3). The completion times of R_1, R_2 and R_3 are 25, 0 , 22 right now. Finally the Third Max ET in R_1 is T_1 and Min CT of T_1 is 23 and on R_2. So T_1 is rescheduled to R_2. Therefore the makespan of whole system is 25 time units. Best-min heuristic is shown at figure 5.

Table 1 Example of A Grid System

	R1	R2	R3
T1	45	23	22
T2	45	70	22
T3	25	63	23

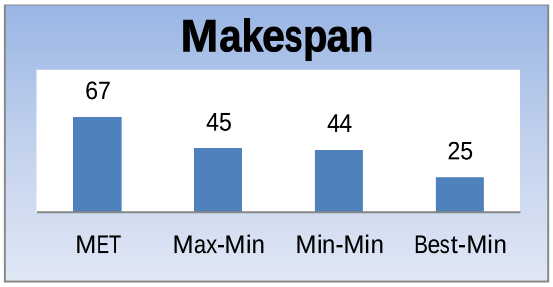

Fig. 3 The Makespan

The resource utilization for all heuristics is shown in figure 6. As you can see, the best-min has the best resource utilization among all using formula (4), the sum of completion times of final jobs is 70, so best-min's resource utilization is 92%. Similarly max-min and min-min resource utilizations' are 69% and 52% respectively.

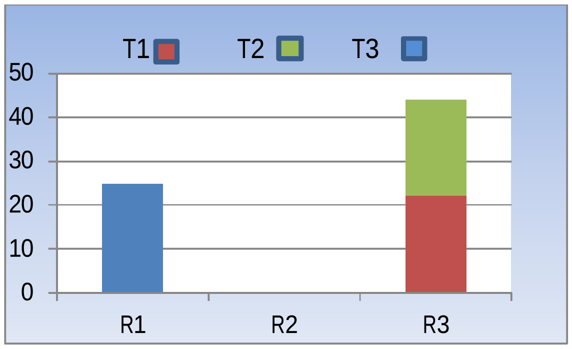

Fig. 4 The Min-min mapping process

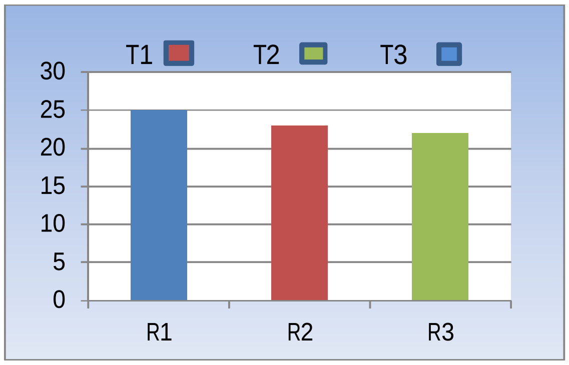

Fig. 5 The Best-min mapping process

The matching proximity amount for MET is always one because according to formula (3) the degree of proximity of a given schedule to the schedule produced by the MET method is evaluated. Min-min produces 0.971 while best-min and max-min respective amounts are 0.957 and 0.766. The matching proximity for four heuristics is shown in figure 7.

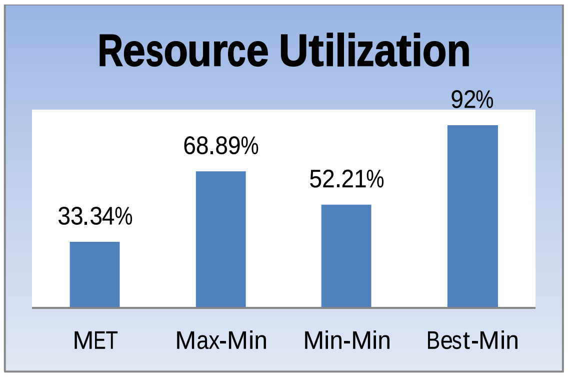

Fig. 6 The Resource Utilization

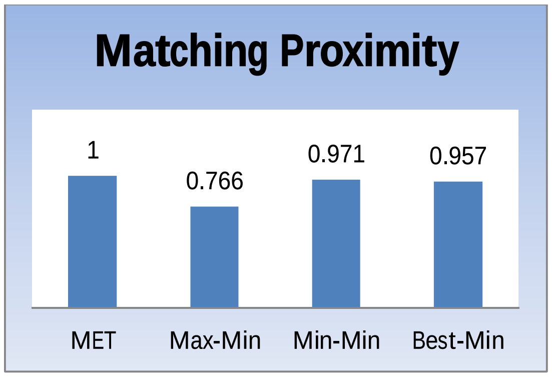

Fig. 7 The Matching Proximity

5- Performance Analysis

For comparison of our proposed heuristic with other scheduling algorithms, ETC model is used as benchmark experiments[5], [6] is used. This model is based on 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. The twelve combinations are shown in Table 2. The amount of variance between the execution times of tasks in the MT for a given machine is defined as task heterogeneity. In environments with high task heterogeneity, different applications with simple, large and complex tasks are submitted to execute in grid system. Machine heterogeneity represents the variation of execution times for a given task across the resources. A grid system containing similar resources is represented as low machine heterogeneity, while high machine heterogeneity represents computing resources of different types. To capture different aspects of realistic mapping situations, different ETC matrix consistencies are defined. An ETC matrix is defined consistent if a machine m_i executes task t faster/slower than machine m_j, then m_i executes all the tasks faster/slower than m_j. Inconsistent ETC matrix means that a machine $m_i$executes some tasks faster than machine m_j and some other tasks slower than m_j or vise versa. The instances are labeled as x_yyzz. X means the type of consistency. The values of x could be c(consistent), i(inconsistent) and s(semi consistent). Yy indicates the task heterogeneity level. Yy could be Hi or Lo. Hi means high heterogeneous, and Lo means low heterogeneous. Zz indicates machine heterogeneity and again it could be Hi or Low. Hi means high heterogeneous, and Lo means low heterogeneous. The combination of ETC instances are shown at table 2.

Table 2. Combinations of Heterogeneity and Consistency

Heterogeneity		Consistency		
Task	Machine	Consistent	Inconsistent	Semi- inconsistent
high	high	c_hihi	i_hihi	s_hihi
high	low	c_hilo	i_hilo	s_hilo
low	high	c_lohi	i_lohi	s_lohi
low	low	c_lolo	i_lolo	s_lolo

A computer program in java language is developed for proposed best-min, max-min and min-min algorithms. The Java program produces respective schedule for tasks and calculates the values of various objective functions explained in section 3.

The makespan of the scheduling algorithms for the twelve different instances of the ETC matrices are shown in table 3. As you can see, best-min algorithm produces minimum makespan in all twelve matrices. min-min is the second algorithm with minimum makespan for all instances and max-min is ranked three.

Geometric mean of all twelve matrices are calculated to show the overall improvement in makespan. Figure 8 shows the geometric mean of makespan for twelve instances.

Figure 9 show the values of resource utilization for the three algorithms. Best-min gives the maximum resource utilization for seven instances and max-min gives the max in other five instances. Min-min is the worst heuristic with the respect to the resource utilization criterion. According to simulation results best-min gives the best resource utilization in all consistence and all semi-consistence (except s-lolo) instances. S-lolo's amounts in best-min and max-min are very close to each other (best-min 97.7% and max-min 97.8%). Also max-min is a bit better (around 1 to 1.5%) in all inconsistence instances. Resource utilization's amounts are also provided in table 4.

Table 3. The makespan results of methods

	i-hihi	**i-hilo**	**i-lohi**	**i-lolo**	**c-hihi**	**c-hilo**	**c-lohi**	**c-lolo**	**s-hihi**	**s-hilo**	**s-lohi**	**s-lolo**
Best-Min	**3174557**	**60140**	**79691**	**2280**	**7906326**	**133645**	**208437**	**4433**	**4043719**	**86238**	**120547**	**2865**
Min-Min	3632431	76357	104746	2665	8243812	148263	268803	4968	4650968	98409	137234	3480
Max-Min	6830643	148852	243027	4732	11348675	205736	410806	6553	8713869	168730	323924	5483

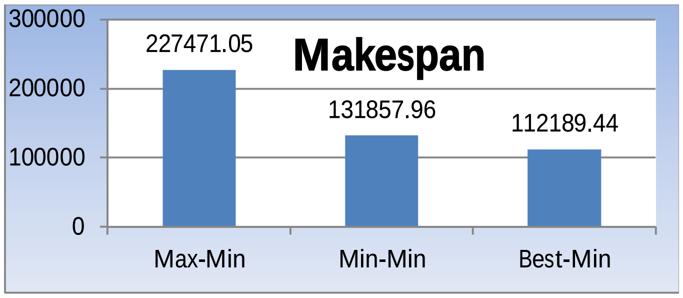

Fig. 8 Comparison Results on makespan

Table 4. The Resource Utilization results of methods

	i-hihi	**i-hilo**	**i-lohi**	**i-lolo**	**c-hihi**	**c-hilo**	**c-lohi**	**c-lolo**	**s-hihi**	**s-hilo**	**s-lohi**	**s-lolo**
Best-Min	0.962	0.935	0.966	0.938	**0.971**	**0.966**	**0.987**	**0.970**	0.977	**0.961**	**0.968**	**0.963**
Min-Min	0.883	0.821	0.877	0.817	0.911	0.879	0.902	0.866	0.875	0.813	0.885	0.829
Max-Min	**0.976**	**0.949**	**0.975**	**0.949**	0.958	0.948	0.961	0.954	**0.978**	0.946	0.967	0.956

Figure 10 shows the obtained matching proximity of the three heuristics for the twelve different instances of the ETC matrices. For eight instances min-min gives the best matching proximity. Best-min gives maximum matching proximity for the remaining four instances while max-min gives the worst amounts in all twelve instances. According to matching proximity simulation results best-min produced better amounts in four consistence instances. That's because of min-min assigns tasks to resource with high processing power especially in consistent matrices.

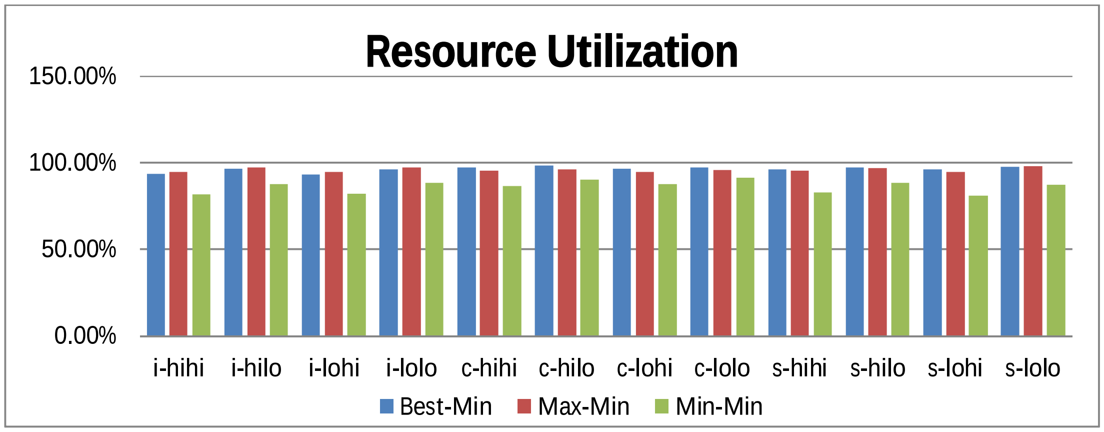

Fig. 9 Comparison Results on Resource Utilization

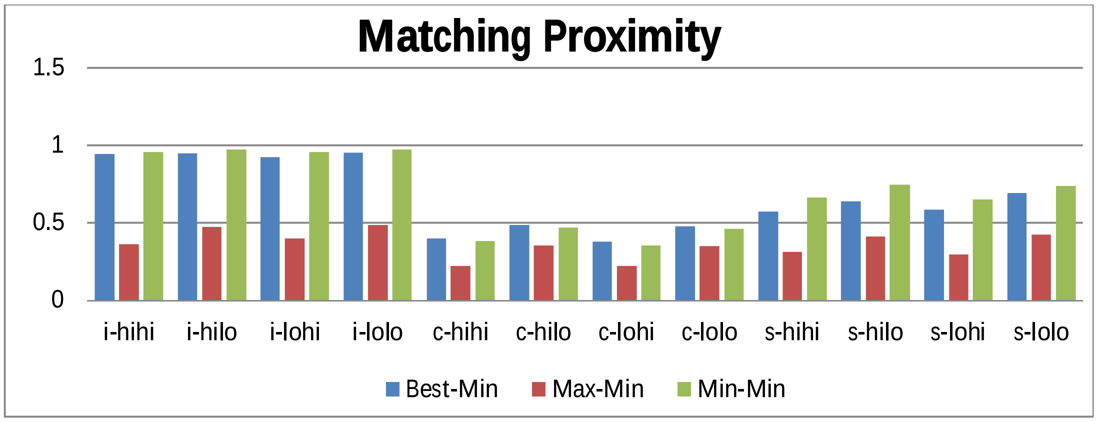

Fig. 10 Comparison results on Matching Proximity

6- Conclusions

Due to the importance of scheduling in Grid computing, presenting a new scheduler that can optimize two or more conflicting objective functions at the same time is promising. Min-min and max-min are the simplest and most well known scheduling algorithms for grid computing. However, when the number of small tasks is more than the number of large tasks in a meta-task, the makespan produced by Min-min becomes big. Furthermore, resource utilization is relatively low in min-min. In contrast max-min gives high resource utilization while fail in producing good makespan results. To overcome these two disadvantages, a new grid scheduling heuristic (best-min) is proposed. Best min is executed in two phase and try to minimize makespan while maximize resource utilization and matching proximity.

REFERENCES

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

[2] Sahu R., Chaturvedi A.K. ,Jan 2011, "Many-Objective Comparison of Twelve Grid Scheduling Heuristics ", International Journal of Computer Applications ", Vol13, pp. 9-17.

[3] Pop F., Dobre C., Cristea V. , jun 2008, "Evaluation of Multi-Objective Decentralized Scheduling for Applications in Grid Environment", IEEE, ICCP, pp.231-238.

[4] Chang H.J., Wu J.J., Liu P., Aug 2009, " Job Scheduling Techniques for Distributed Systems with Heterogeneous Processor Cardinality", 10th International Symposium on Pervasive Systems, Algorithms, and Networks , pp.57-62.

[5] Braun T.D., Siegel H.J., Beck N., 2001, "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, pp.810-837.

[6] Braun T., Siegel H., Beck N., Boloni L., Maheshwaran M., Reuther A., Robertson J., Theys M., Yao B., Hensgen D., R.Freund, 1999, "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.

[7] Izakian H., Ajith A., Vaclav S., 2009, " Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments", International Workshop on HPC and Grid Applications(IWHGA2009), China, IEEE Press, pp. 8-12.

[8] Izakian H., Ajith A., Vaclav S., 2009, "Performance Comparison of Six Efficient Pure Heuristics for Scheduling Meta-Tasks on Heterogeneous Distributed Environments" , Journal of Neural Network World, Volume 19, Issue 6, pp. 695-710.

[9] Xhafa F., Carretero J., Barolli L., Durresi A., 2007, "Immediate Mode Scheduling In Grid Systems", International Journal Of Web & Grid Services, Volume 3, No 2, pp219-236.

[10] Xhafa F., Barolli L., Durresi A., 2007, "Batch Mode Scheduling In Grid Systems", International Journal Of Web & Grid Services, Volume 3, No 1, pp19-37.

[11] Maheswaran M., Ali S., Siegel H.J., Hensgen D., Freund R.F., 1999, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems", Journal of Parallel and Distributed Computing 59,Volume 9, Issue 3, pp 107-131.

[12] Meihong W., Wenhua Z. , 2010, "A comparison of four popular heuristics for task scheduling problem in computational grid ", 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM) ,IEEE, pp 1-4.

[13] Alharbi F., May 2012, "Multi Objective Heuristic Algorithm For Grid Computing ", International Journal Of Computer Applications, Vol 46, Issue 18, pp 39-45.

[14] Yu K.M., Chen C.K., 2008, "An Adaptive Scheduling Algorithm for Scheduling Tasks in Computational Grid", in IEEE Seventh International Conference on Grid and Cooperative Computing, pp185-189.

[15] Ibarra O.H., Kim C.E., Apr 1977, "Heuristic algorithms for scheduling independent tasks on non identical processors", Journal of Association for Computing Machinery, Vol.24, Issue.2, , pp280-289.