Intermediate Mode Scheduling in Computational Grid

Sanjaya Kumar Panda[1], Pratik Agrawal[2] and Durga Prasad Mohapatra[3]
[1,3] Department of Computer Science and Engineering
[1] Indian School of Mines Dhanbad, India, [2]IBM Bangalore, India, [3]National Institute of Technology Rourkela, India
[1] sanjayauce@gmail.com, [2] pratikag99@gmail.com, [3] durga@nitrkl.ac.in

***Abstract*—Mode of Scheduling plays the key role in Grid Scheduling. It is of two types, immediate and batch mode. Immediate mode takes one by one task in a sequence. But the batch mode takes in a random sequence. So, task assignment is mainly based on the mode selection. The task may be assigned to the resource as soon as arrive or in a batch. In this paper, we have introduced a new mode of heuristic called as intermediate mode (or Multi-ζ batch mode). This mode considers the random arrival of task in a multi-batch sequence. Alternatively, arrivals of tasks are unknown in this mode. Here, we have taken a range of task arrival for simplicity. This mode is introduced to be a part of the real life aspects. The two existing batch mode heuristics: Min-Min and Max-Min are experimented with intermediate mode scheduling. We have taken two performance measures, makespan and resource utilization to evaluate the performance.**

Keywords—Immediate Mode; Batch Mode; Intermediate Mode; Min-Min; Max-Min; Scheduling; Benchmark Data Set

I. INTRODUCTION

A grid is an emerged area to share resources, aggregate resources or solve large scale complex problems [1]. It is decentralized, heterogeneous and distributed in nature. Resources are distributed geographically and it may be owned, processed and organized by different domains [2] [3]. So, it may join or leave at any point of time. So, Grid Referral Service (GRS) keeps track of all information about resources [2].

Jobs are submitted from different grid user to Grid Resource Broker (GRB). A user may have a different set of characteristics like types of jobs, modes of computation e.g. preemptive or non-preemptive, minimization strategy e.g. cost, time, or resource utilization [2]. While resource may have characteristics like number of processors, mode of processing e.g. immediate or batch mode, speed of processing, load balancing factor e.g. overload factor or under load factor and the cost of processing [2]. Jobs are divided into a number of smaller units called tasks. Then, Tasks are processed by one or more domains. Finally, the results are combined together to form a single unit. But the user does not have any record that in which resources the tasks are completed. This whole scenario looks like a single system. So, it is referred as a Single System Image (SSI). It hides the heterogeneous and distributed nature of the entire system [4].

Task scheduling in heterogeneous environment and mapping the best task-resource pair is a NP-Complete problem [5-6]. It is not possible to get a polynomial time complexity heuristic. So many heuristics are proposed in the last decades. These heuristics are giving results based on different constraints. It may be based on QoS parameter, bandwidth, priority, threshold, fault tolerance or trust [7-12].

Mode of scheduling manages the computation sequence of the tasks. However, GRB is responsible to map the tasks and the resources. It asks to GRS about resource availability as well as all information about a resource. It assigns a task to a resource based on the availability. Also it schedules the task as soon as arrive or in a batch. In this paper, we assume that the tasks are independent of each other. So, tasks can be scheduled to the any available resources at any point of time.

Mode of scheduling is of two types, immediate and batch mode. Immediate mode serves the request as soon as arrive in First Come First Serve (FCFS) basis. But the batch mode serves the request after the arrival of a set of tasks in a random sequence. Based on these modes, we have introduced intermediate mode scheduling (or Multi-ζ batch mode). In this mode, task arrivals are processed in a random sequence. In other words, it supports online arrival of task. The GRB has no idea about how many job computed per batch. We have applied the existing batch mode heuristics e.g. Min - Min and Max - Min scheduling into the proposed mode of scheduling.

The remainder of this paper is organized as follows. In section II, we explain related works. In section III, we have a brief introduction about various modes of scheduling. Subsequently, we introduce the new mode of scheduling in section IV. Experimental results are shown in section V. Finally, we conclude this paper in section VI.

II. RELATED WORK

Many scheduling heuristics are proposed in last two decades. The heuristics are fall into either immediate or batch mode types. In this section, we will discuss some related works which closely focused on our work.

Maheswaran et al. have presented two immediate mode and one batch mode heuristics [5]. The immediate mode heuristics are Switching Algorithm (SA) and K-Percent Best (KPB). These heuristics are compared with the two well-known existing heuristics Minimum Execution Time (MET) and Minimum Completion Time (MCT). The batch mode heuristic is called sufferage heuristic. It is compared with two existing heuristics, Min - Min and Max - Min.

Ali et al. have focused on the characteristics of the matrices [6]. Here, Coefficient-of-Variation-Based (CVB) method has been proposed.

Braun et al. have compared the eleven static heuristics [13]. It is shown that Min - Min heuristic outperforms than other heuristics like Opportunistic Load Balancing (OLB), MET, MCT, Max - Min, duplex, Genetic Algorithm (GA), simulated annealing, tabu and A^*.

Xhafa et al. [14] have implemented and compared the five immediate mode heuristic using the benchmark data set proposed by Braun et al. [13]. Again, Xhafa et al. [15] have implemented five batch mode heuristic. The relative cost and Longest Job to Fastest Resource – Shortest Job to Fastest Resource (LJFR-SJFR) has added in addition to Braun et al. [13].

Chaturvedi et al. have presented a new heuristic which consist of two stages [16]. First, MET is applied to the instance. Second, the overloaded resource transfer tasks to the under load resource. It manages the load imbalance problem. Mostly, this heuristic outperforms in consistency type of instances.

Abuelenin et al. have presented three batch mode scheduling in the context of trust [17]. It shows that the trust model has minimized the makespan as well as maximized the resource utilization.

Panda et al. and Etminani et al. have proposed a new selective heuristic based on Semi-Interquartile (SI) and standard deviation respectively [18-20]. Both heuristic chooses either Min-Min or Max-Min heuristic in iteration.

III. Preliminaries

In this section, we have discussed the existing mode of heuristic along with merits and demerits. Before that notations and their definitions are presented.

A. *Notations Used and Their Definitions*

Notation	Definition
α	Number of tasks
β	Number of resources
τ	Range of number of tasks arrival
ζ	Number of tasks arrival
TQ	Task Queue
ETC	Expected Time to Compute
RU	Resource Utilization

B. *Heuristics*

In the grid, heuristics are categorized into two types: immediate mode and batch mode.

1) *Immediate Mode*

In immediate mode, the tasks are computed one after another. Alternatively, the task arrives first in TQ, will be computed first. MET and MCT are the heuristics for immediate mode scheduling. Normally, this mode takes ζ value as one. Even if, $\zeta > 1$, it computes one task at a time. Performance of immediate mode may be enhanced when $\zeta > 1$because it has more than one task. When $\zeta > 1$, it selects the first arrived task rather than the best task available in the instance. It leads to increase the makespan. We present the pseudo code of immediate mode as follows.

```
1.  begin
2.      Add |ζ| to TQ
3.      if |ζ| = NULL
4.          TQ = NULL
5.      else
6.          Remove task $T_I$ from TQ
            // $T_I$ = Task in the head of the queue
7.          Assign task $T_I$ to resource $R_j$
            // $R_j$ = Assign $j^{th}$ resource as per heuristic.
8.      end if
9.  end
```

Immediate mode adds the tasks to the TQ in iteration. Then, it removes one task from the head of the queue and assigns the task to the resource based on the immediate mode heuristic. It iterates until all the tasks are successfully mapped (i.e. TQ is empty).

2) *Batch Mode*

In batch mode, all tasks are arriving at a time. One task is selected from the batch of the tasks. Alternatively, the task arrives first in TQ, may/may not be computed first. Min-Min and Max-Min are the heuristics for batch mode scheduling. This mode of heuristic takes ζ value as size of tasks in a batch. If $\zeta = 1$, batch mode heuristic acts like immediate mode heuristic. In real life, batch of tasks may not be arrived at a time. Note that, it always finds the best task from the batch of tasks. We present the pseudo code of batch mode as follows.

```
1.  begin
2.      Add |α| to TQ
3.      if |α| = NULL
4.          TQ = NULL
5.      else
6.          for 1 to |α|
7.              Find task $T_i$ from TQ
8.              // $T_i = i^{th}$ task as per heuristic, $1 < i < |\alpha|$
9.              Assign task $T_i$ to resource $R_j$
10.             // $R_j$ = Assign $j^{th}$ resource as per heuristic
11.             Remove task $T_i$ from TQ
12.         end for
13.     end if
14. end
```

Batch mode adds the tasks to the TQ after $|\alpha|$ tasks are successfully mapped. It removes one task from the queue and assigns the task to the resource based on the batch mode heuristic. It iterates until all the tasks are successfully mapped (i.e. TQ is empty).

IV. Intermediate Mode

By considering the merits and demerits of immediate mode and batch mode, we have proposed intermediate mode heuristic in this paper.

A. Description

Intermediate mode heuristic is a variation of batch mode heuristic. The value of ζ varies from 2 to α-1. If ζ = 1, intermediate mode heuristic acts like immediate mode heuristic. If ζ = α, it acts like batch mode heuristic. Equation 1 shows the selection of heuristic based on different value of ζ. Note that, α is unknown in intermediate mode.

$$\text{Heuristic} = \begin{cases} \text{Immediate Mode} & \text{if } \zeta = 1 \\ \text{Intermediate Mode} & \text{if } 1 < \zeta < \alpha \\ \text{Batch Mode} & \text{if } \zeta = \alpha \end{cases} \quad (1)$$

B. Proposed Mode

In this mode, we have taken a random function to determine the ζ value. In each iteration, ζ value is determined. Based on ζ value, numbers of tasks are computed. The ζ values of each instance are shown in Table I. For 512 × 16 instances, the values of ζ are 58, 46, 62, 38, 40, 36, 36, 60, 51, 50 and 35. It means 58 tasks are going to be executed in the first iteration, 46 tasks are going to be executed in second iteration and so on. The range of intermediate mode scheduling is shown in Equation 2.

$$\left\lfloor \frac{\alpha}{\beta} \right\rfloor \le \tau \le \left\lfloor \frac{\alpha}{\left(\frac{\beta}{2}\right)} \right\rfloor$$

$$= \left\lfloor \frac{\alpha}{\beta} \right\rfloor \le \tau \le \left\lfloor \frac{2\alpha}{\beta} \right\rfloor \quad (2)$$

Let us consider α = 512 and β = 16. So, the range of intermediate mode is shown in Equation 3.

$$\left\lfloor \frac{512}{16} \right\rfloor \le \tau \le \left\lfloor \frac{512}{\left(\frac{16}{2}\right)} \right\rfloor$$

$$= \left\lfloor \frac{512}{16} \right\rfloor \le \tau \le \left\lfloor \frac{2 \text{ X } 512}{16} \right\rfloor$$

$$= 32 \le \tau \le 64 \quad (3)$$

In this paper, we have taken τ value as 32 to 64 for simplicity. In real life situation, it may vary.

TABLE I. TASK ARRIVALS PER INSTANCES.

Size of instances	Value of ζ
512 × 16	58, 46, 62, 38, 40, 36, 36, 60, 51, 50, 35
1024 × 32	58, 46, 62, 38, 40, 36, 36, 60, 51, 50, 36, 60, 52, 43, 48, 45, 34, 39, 36, 38, 39, 45, 32
2048 × 64	58, 46, 62, 38, 40, 36, 36, 60, 51, 50, 36, 60, 52, 43, 48, 45, 34, 39, 36, 38, 39, 45, 33, 61, 63, 48, 48, 43, 61, 44, 35, 57, 44, 39, 45, 35, 36, 63, 63, 50, 33, 39, 43, 59, 14

We have taken 512 × 16 instances from Braun et al. [13]. So, we have assumed that α = 512 and β = 16. As we have considered this data set, the upper limit is known to us. But in real life (or intermediate mode), there is no such limit. We present the pseudo code of intermediate mode as follows.

```
1.  begin
2.      Calculate $\left\lfloor \frac{\alpha}{\beta} \right\rfloor \le \tau \le \left\lfloor \frac{2\alpha}{\beta} \right\rfloor$
3.      Choose $\zeta$ from the range of $\tau$
4.      Add $|\zeta|$ to TQ
5.      if $|\zeta|$ = NULL
6.          TQ = NULL
7.      else
8.          for 1 to $|\zeta|$
9.              Find task $T_i$ from TQ.
                // $T_i$ = $i^{th}$ task as per heuristic, $1 < i < |\zeta|$
10.             Assign task $T_i$ to resource $R_j$
                // $R_j$ = Assign $j^{th}$ resource as per heuristic.
11.             Remove task $T_i$ from TQ.
12.         end for
13.     end if
14. end
```

Intermediate mode finds the |ζ| value from the range of tau. It adds the tasks to the TQ after |ζ| tasks are successfully mapped. It removes one task from the queue and assigns the task to the resource based on the heuristic. It iterates until all the tasks are successfully mapped (i.e. TQ is empty). The value of |ζ| varies from iteration to iteration. It is our main contribution in this paper.

Let $|\zeta_1|$ be the total number of tasks computed in the first iteration, $|\zeta_2|$ be the total number of tasks computed in the second iteration and $|\zeta_n|$ be the total number of tasks computed in n^{th} iteration. Then the value of α is the sum of the tasks computed in all iterations. It is shown in Equation 4.

$$\alpha = \sum_{i=1}^{n} |\zeta_i| \quad (4)$$

V. EXPERIMENTAL RESULTS

In this section, we present the experimental results of the intermediate mode scheduling. We have taken Braun et al. [13] data set (or instances) to evaluate the proposed mode scheduling.

A. Data Set

The instances of Braun et al. are categorized into 3 sub-types. They are consistent, inconsistent and semi-consistent. The general forms of these instances are u_t_jjrr where u indicates the uniform distribution. The t indicates the type of consistency, jj shows the task heterogeneity and rr shows the resource heterogeneity. The value of jj and rr is either hi or lo. Each type of consistency having following combinations, hihi, hilo, lohi and lolo. So, we have 12 different types of ETC instances. Figure 1 shows the categorization Braun et al. instances.

A matrix is said to be consistent if a resource β_i takes less time for task α_i, in compare to the resource β_j, then for all tasks, resource β_i takes least time than resource β_j. In inconsistent metrics, task α_i may take least time in resource β_i while other tasks may take least time in other resources. It means tasks are not taking least time in a particular resource. Semi-consistent matrix holds a consistent subset matrix.

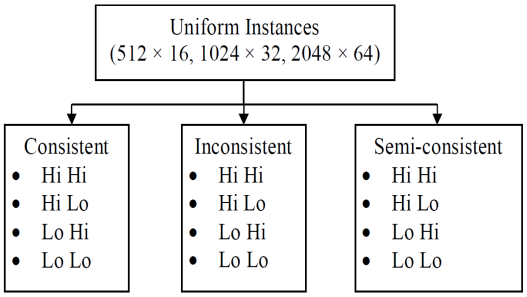

Figure 1. Categorization of Braun et al. instances.

B. *Cases*

We have considered three cases, 512 × 16, 1024 × 32 and 2048 × 64. In each case, row indicates the number of tasks and column indicates the number of resources. Here, each case consists of 12 different types of matrices.

C. *Illustrations*

Let us consider an example consists of 512 tasks and 16 resources. Assume that, u_c_hihi instance is taken. It means uniform distribution with consistent matrix. It has high task heterogeneity and high resource heterogeneity. Note that, total numbers of tasks are unknown in intermediate scheduling. We have assumed that the range of task arrival (τ) is 32 to 64 (both inclusive). So, we use a random function to determine the task arrival (ζ). According to the random function, we get 58, 46, 62, 38, 40, 36, 60, 51, 50 and 35 respectively. If we run the random function again, we may not get the above sequence. So, it is taken in a time instant. First, 58 tasks are computed in intermediate mode scheduling. This mode of heuristic has no knowledge of the sequence given after 58. Second, 46 tasks are computed and so on. Finally, we get the makespan value as 1.0166×10^7.

In each instance, we have considered the same random function sequence. In this paper, we have applied intermediate mode scheduling over two existing batch mode heuristics such as Min-Min and Max-Min. We can apply intermediate mode in immediate mode heuristics. But it gives the same result as immediate mode because the tasks are computed one after another.

D. *Results*

We have simulated the heuristics using MATLAB R2010b version 7.11.0.584. Makespan and Resource Utilization (RU) of min-min heuristic in intermediate mode are shown in Table II and Table III respectively. Makespan and RU of max-min heuristic in intermediate mode are shown in Table IV and Table V respectively. Each heuristic is tested under 512 × 16 (Case 1), 1024 × 32 (Case 2) and 2048 × 64 (Case 3) data sets. Graphical representation of makespan value (Min-min) is shown in Figure 2. Graphical representation of makespan value (Max-min) is shown in Figure 3.

TABLE II. MAKESPAN VALUES OF INTERMEDIATE MODE SCHEDULING (MIN-MIN).

Instance	Makespan (Case 1)	Makespan (Case 2)	Makespan (Case 3)
u_c_hihi	1.0166×10^7	2.8030×10^7	2.5465×10^7
u_c_hilo	1.7060×10^5	2.8572×10^6	2.4457×10^6
u_c_lohi	3.2393×10^5	2.7269×10^3	2.5476×10^3
u_c_lolo	5.8218×10^3	2.8531×10^2	2.4985×10^2
u_i_hihi	3.9365×10^6	7.0648×10^6	3.5620×10^6
u_i_hilo	8.5327×10^4	6.6892×10^5	3.9054×10^5
u_i_lohi	1.3295×10^5	7.2203×10^2	3.6930×10^2
u_i_lolo	2.9201×10^3	6.9890×10^1	4.0690×10^1
u_s_hihi	5.5299×10^6	1.7980×10^7	1.5191×10^7
u_s_hilo	1.1047×10^5	1.6746×10^6	1.3569×10^6
u_s_lohi	1.7105×10^5	1.7268×10^3	1.4208×10^3
u_s_lolo	4.0299×10^3	1.7670×10^2	1.5070×10^2

TABLE III. RESOURCE UTILISATION VALUES OF INTERMEDIATE MODE SCHEDULING (MIN-MIN).

Instance	RU (Case 1)	RU (Case 2)	RU (Case 3)
u_c_hihi	0.9256	0.9105	0.9150
u_c_hilo	0.9541	0.9200	0.9274
u_c_lohi	0.9324	0.9007	0.9155
u_c_lolo	0.9384	0.9130	0.9334
u_i_hihi	0.9114	0.9041	0.8763
u_i_hilo	0.9645	0.8993	0.8659
u_i_lohi	0.9222	0.8878	0.9203
u_i_lolo	0.9617	0.8914	0.8260
u_s_hihi	0.9309	0.8910	0.8634
u_s_hilo	0.9453	0.9003	0.8907
u_s_lohi	0.8860	0.8641	0.8707
u_s_lolo	0.9438	0.8492	0.8588

TABLE IV. MAKESPAN VALUES OF INTERMEDIATE MODE SCHEDULING (MAX-MIN).

Instance	Makespan (Case 1)	Makespan (Case 2)	Makespan (Case 3)
u_c_hihi	1.2800×10^7	3.5985×10^7	3.1587×10^7
u_c_hilo	2.0339×10^5	3.6091×10^6	3.0689×10^6
u_c_lohi	4.1913×10^5	3.5161×10^3	3.2692×10^3
u_c_lolo	6.8891×10^3	3.7003×10^2	3.0880×10^2
u_i_hihi	5.5328×10^6	8.8184×10^6	4.0519×10^6
u_i_hilo	1.1861×10^5	8.1415×10^5	4.3581×10^5
u_i_lohi	1.8903×10^5	8.5709×10^2	4.1651×10^2
u_i_lolo	3.9876×10^3	8.3320×10^1	4.4050×10^1
u_s_hihi	8.1120×10^6	2.1557×10^7	1.7156×10^7
u_s_hilo	1.4574×10^5	1.9977×10^6	1.5633×10^6
u_s_lohi	2.3635×10^5	2.0091×10^3	1.6200×10^3
u_s_lolo	5.3405×10^3	2.2031×10^2	1.7633×10^2

TABLE V. RESOURCE UTILISATION VALUES OF INTERMEDIATE MODE SCHEDULING (MAX-MIN).

Instance	RU (Case 1)	RU (Case 2)	RU (Case 3)
u_c_hihi	0.9962	0.9736	0.9619
u_c_hilo	0.9941	0.9763	0.9630
u_c_lohi	0.9905	0.9560	0.9696
u_c_lolo	0.9900	0.9820	0.9772
u_i_hihi	0.9747	0.9523	0.8540
u_i_hilo	0.9956	0.9193	0.8977
u_i_lohi	0.9682	0.9551	0.9062
u_i_lolo	0.9788	0.9429	0.8610
u_s_hihi	0.9397	0.9684	0.9265
u_s_hilo	0.9848	0.9591	0.9403
u_s_lohi	0.9863	0.9598	0.9393
u_s_lolo	0.9799	0.9349	0.9141

VI. CONCLUSION

In this paper, we have presented a new mode of scheduling, called as intermediate mode scheduling. The main goal of this scheduling is to handle the real life tasks. It acts like batch mode in each iteration. But in each iteration, numbers of tasks are different. The two batch mode heuristics, min-min and max-min have been applied to the new mode of scheduling by using Braun et al. [13] benchmark data sets. The performances of these heuristics are shown in terms of makespan and resource utilization.

In future, we can extend it to fault-tolerant scheduling, security and predictive based scheduling. As resources are distributed geographically, it may be leave in the mean time because of numerous types of fault. However, the scheduler may predict the types of task arriving, QoS parameters and many more. So that prediction based scheduling can also be introduced. We must focus on the above aspects, to propose a new dynamic and robust scheduling.

REFERENCES

[1] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of High Performance Computing Applications, Vol. 15, No. 3, pp. 200-222, 2001.

[2] M. Murshed and R. Buyya, "Using the GridSim Toolkit for Enabling Grid Computing Education", International Conference on Communication Networks and Distributed Systems Modeling and Simulation, 2002.

[3] A. Abraham, R. Buyya and B. Nath, "Natures Heuristics for Scheduling Jobs on Computational Grids", Eighth IEEE International Conference on Advanced Computing and Communications, 2000.

[4] R. Buyya, T. Cortes and H. Jin, "Single System Image (SSI)", The International Journal of High Performance Computing Applications, Vol. 15, No. 2, pp. 124-135, 2001.

[5] 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", Journal of Parallel and Distributed Computing, Vol. 59, No. 2, 107-131, 1999.

[6] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen and S. Ali, "Task Execution Time Modeling for Heterogeneous Computing Systems", Heterogeneous Computing Workshop, pp. 185-199, 2000.

[7] H. Xiaoshan, X. H. Sun and G. V. Laszewski, "QoS Guided Min-Min Heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, Vol. 18, No. 4, pp. 442-451, 2003.

[8] S. Abuelenin, "Trust Based Grid Batch Mode Scheduling Algorithms", The 8th International Conference on INFOrmatics and Systems, pp. 46-54, 2012.

[9] S. K. Panda and P. M. Khilar, "A Two-Step QoS Priority for Scheduling in Grid", Second IEEE International Conference on Parallel, Distributed and Grid Computing, pp. 502-507, 2012.

[10] S. K. Panda, "TBS: A Threshold Based Scheduling in Grid Environment.", Proceedings of 3rd IEEE International Advance Computing Conference, pp. 22-26, 2013.

[11] S. K. Panda, P. M. Khilar and D. P. Mohapatra, "FTM^2: Fault Tolerant Batch Mode Heuristics in Computational Grid", 10th International Conference on Distributed Computing and Internet Technology, Springer, Vol. 8337, pp. 98-104, 2014.

[12] S. K. Panda, P. M. Khilar and D. P. Mohapatra, "FTMXT: Fault Tolerant Immediate Mode Heuristics in Computational Grid", International Conference on Informatics and Communication Technologies for Societal Development, Springer, 2014.

[13] T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys and B. Yao, "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.

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

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

[16] A. K. Chaturvedi and R. Sahu, "New Heuristic for Scheduling of Independent Tasks in Computational Grid", International Journal of Grid and Distributed Computing, Vol. 4, No. 3, pp. 25-36, 2011.

[17] S. Abuelenin, "Trust Based Grid Batch Mode Scheduling Algorithms", The 8th International Conference on INFOrmatics and Systems, pp. 46-54, 2012.

[18] S. K. Panda, S. K. Bhoi and P. M. Khilar, "A Semi-Interquartile Min-Min Max-Min (SIM^2) Approach for Grid Task Scheduling", International Conference on Advances in Computing, Springer, Vol. 174, pp. 415-421, 2012.

[19] S. K. Panda, "Effcient Scheduling Heuristics for Independent Tasks in Computational Grids", National Institute of Technology, Rourkela, M. Tech. Thesis, 2013.

[20] K. Etminani and M. Naghibzadeh, "A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling", 3rd IEEE/IFIP International Conference on Internet, 2007.

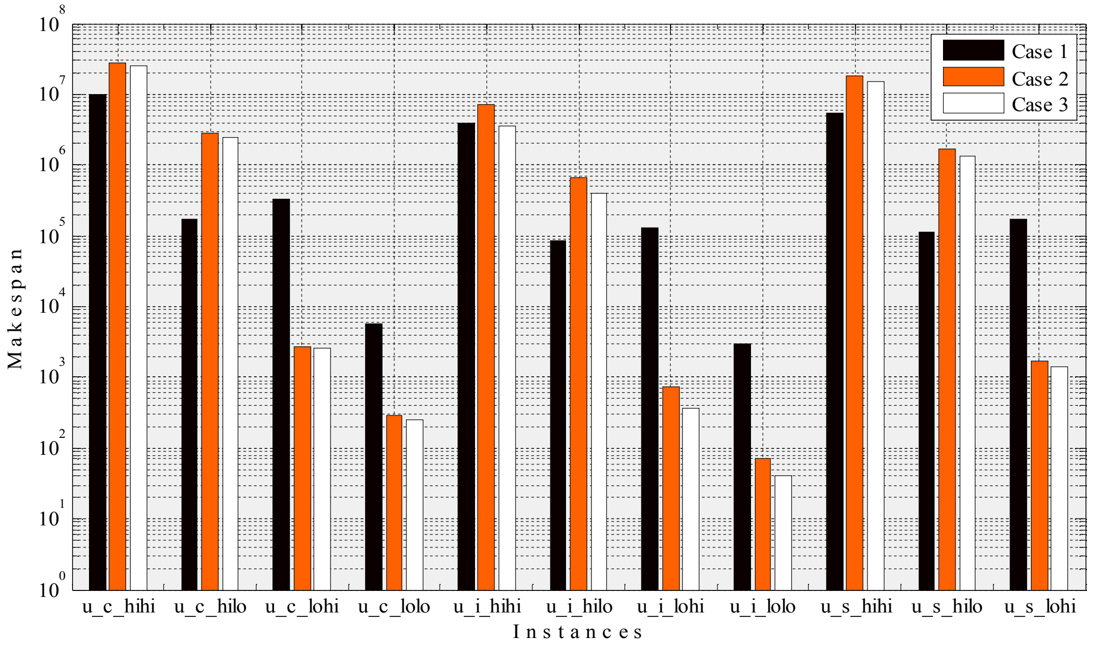

Figure 2. Graphical representation of makespan value (Min-Min).

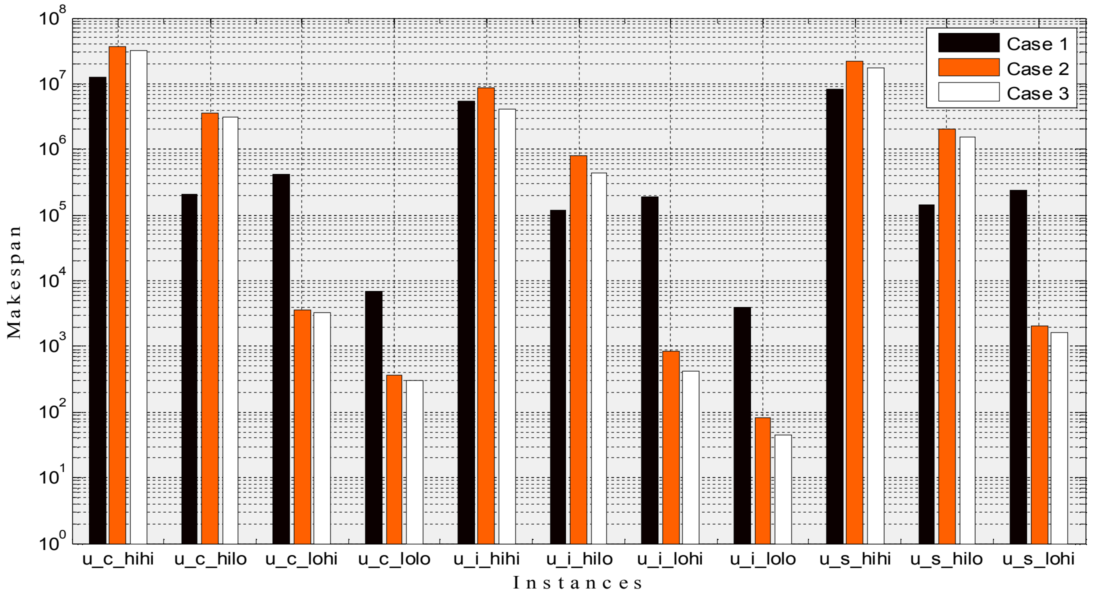

Figure 3. Graphical representation of makespan value (Max-Min).