X-DualMake: Novel Immediate Mode Scheduling Heuristics in Computational Grids

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*—Scheduling is an important aspect in grid computing. Now-a-days, the computational grids are the important platform for job scheduling. The performance of the computational grids can be improved using an efficient scheduling heuristic. In job scheduling, a user submits the job to the grid resource broker. Then the broker is responsible for dividing the job into a number of tasks. Moreover, it also maps the tasks and the resources to find the perfect match. The primary goal of scheduling is to minimize the processing time and maximize the resource utilization. In this paper, we have proposed three immediate mode heuristics such as First-DualMake, Best-DualMake and Worst-DualMake (named as X-DualMake). These heuristic are scheduled based on the resource idle time. We have also presented five existing heuristics such as MET, MCT, OLB, KPB and SA. The eight heuristics are simulated and the experimental results are discussed. The heuristics are compared using two performance measures makespan and resource utilization.**

Keywords—Grid Computing; Scheduling; Task; Resource; Immediate Mode; Grid Resource Broker; Makespan; Data Set

I. INTRODUCTION

In recent years, grid computing provides high performance computing solution for many scientific or critical applications. The grid computing system is loosely coupled and message passing distributed system where computing resources are autonomous and communication between nodes are performed by passing messages using a high speed interconnection network [1]. The resources are not sharing the clock, memory, bus and peripherals. They are having their local private memory. It is of two types, homogeneous and heterogeneous. Two homogeneous resources can have the same operating system, architecture, peripherals etc. But two heterogeneous resources can have different operating system, architecture, peripherals etc [2]. For instance, one resource has a Linux operating system while other resources may have Windows or Unix operating system.

Grid computing is heterogeneous in nature. It is also decentralized architecture. Message Passing Interface (MPI) and Parallel Virtual Machines (PVM) allow the network of heterogeneous resources to make a huge computational power and storage [3-4]. Grid computing is used to solve large scale complex problem. The problem is divided into a set of smaller sub-problems. Each sub-problem is solved by a heterogeneous resource. At last, the sub-solutions are grouped together to form a unified solution. In other words, the user submits the job to the grid and gets back results without the knowledge about the grid. It is referred as a Single System Image (SSI) [5]. It is an illusion to user. Note that, a user does not have any record about the job and resource assignment. Sharing is not limited to the file, but it can be extended to hardware, software, storage, computation power and many more. But sharing in a company or an organization is known as a virtual organization [6].

The scheduling components are task, resource, Grid Resource Broker (GRB) and Grid Referral Service (GRS) [7-8]. GRS is maintaining a list of resources. But GRB schedules the task to the resource based on the scheduling strategy. It is also a mapping module which map the task and the resource. The goal of scheduling is to minimize the scheduling length (or makespan) and maximize the resource utilization. However, scheduling in the heterogeneous grid environment is a NP-Complete problem [9, 10-13]. For this we need heuristic that gives close to optimal solution. Researchers are coming up with different heuristics to improve over existing approaches. The details of these heuristic are discussed in Section III.

The remainder of the paper is organized as follows: Section II presents the related work. Section III presents a survey on the immediate mode heuristic with an example. Section IV proposes three immediate mode heuristic: First-DualMake (F-DM), Best-DualMake (B-DM) and Worst-DualMake (W-DM). Section V shows the simulation results. This section compares the eight heuristics. We conclude this paper in Section VI.

II. BACKGROUND AND RELATED WORK

Let $T = \{T_1, T_2,..., T_t\}$ indicates a set of t independent tasks and $R = \{R_1, R_2,..., R_r\}$ indicates a set of r resources. We assume that the tasks are arriving one after another in numeric order. The unit of each task is in seconds. The problem is to map the task to the resource which minimizes the total processing time and maximizes the resource utilization.

Maheswaran et al. and Xhafa et al. have discussed five immediate mode heuristics [12, 14]. These heuristics are implemented in discrete event simulator. The heuristics are compared using Braun et al. benchmark instances [10]. It is stated that MCT is a benchmark in the immediate mode heuristic [12]. Apart from the above heuristics, Braun et al. have discussed some more heuristics such as duplex, genetic algorithm, tabu, simulated annealing and A* heuristic and these heuristics are implemented in an interactive software application [10]. It is also stated that min-min heuristic performs better than all other heuristics (i.e. both immediate and batch mode).

Scheduling has two phases: resource selection (matching) and ordering the tasks (scheduling). When a task has arrived, the GRB finds the makespan of each resource. The task is assigned to the resource having less makespan [15]. Rasooli et al. have introduced two rules for matching and three rules for scheduling [15]. Grosan, Dail et al. and Chin et al. have proposed various scheduling applications based on job shop scheduling, mesh based application and list scheduling respectively [16-18].

Quality of service (QoS) refers to bandwidth, speed, memory etc. QoS min-min heuristic combines the QoS and min-min heuristic in which meta tasks are divided into high QoS and low QoS resource. However, the low QoS tasks can be executed in both low QoS and high QoS resource [19]. Many batch mode heuristics have been proposed in grid environment [13, 20-28]. However, the idle time of resources is not considered so far.

III. Scheduling Algorithms

There are many immediate mode scheduling heuristics in computational grids. The heuristics are listed as follows.

A. *Minimum Execution Time* (*MET*) *Heuristic*

It is also known as limited best assignment (LBA) [9]. It assigns the task to the resource which takes the least execution time. If two resources are taking same execution time, then one of the resources is selected randomly. It serves the task at First In First Out (FIFO) basis. As soon as a task arrives, it schedules the task to the corresponding resource. This heuristic is not considering the resource ready time. So, the task may be scheduled to the resource which has already overloaded. It causes a load imbalance problem. It also suffers from the least resource utilization problem. Each task takes O(r) time to find a resource [12] where r represents number of resources.

TABLE I. A 4 × 4 Expected ET Matrix

Task / Resource	R_1	R_2	R_3	R_4
T_1	20	54	49	97
T_2	74	105	81	93
T_3	33	87	58	40
T_4	51	76	69	123

Let us consider an example shown in Table I. There are four tasks <T_1, T_2, T_3 and T_4> and four resources <R_1, R_2, R_3 and R_4>. Assume that all the resources are idle. Task T_1 has 20 time units in resource R_1. Also it has least execution time in resource R_1. So, Task T_1 is assigned to resource R_1. Like Task T_1, Task T_2 has least execution time in resource R_1. So, it is also assigned to resource R_1. As we notice here, even if other resources are idle, it has not considered that resource. It leads to load imbalance problem. Similarly, Task T_3 and Task T_4 are assigned to resource R_1. The makespan is 178.

B. *Minimum Completion Time* (*MCT*) *Heuristic*

It assigns the task to the resource which takes the least completion time. If two resources are taking same completion time, then one of the resources is selected randomly. Completion time is the sum of execution time (ET) and ready time (RT). It is shown in Equation 1. Unlike MET, MCT heuristic considers the resource ready time. Generally, it gives better results than MET heuristic. In MCT, load imbalance is reduced to some extent. This heuristic does not assign the task to the overloaded resource. It means the task may not map to least execution time resource. Each task takes O(r) time to find a resource [12].

$$\text{Completion Time} = \text{Execution Time} + \text{Ready Time} \quad (1)$$

Let us consider the same example shown in Table I. The ready time is initially set to zero. Task T_1 has 20 time units in resource R_1, 54 time units in resource R_2, 49 time units in resource R_3 and 97 time units in resource R_4. The least completion time is taken by resource R_1. So, it is assigned to resource R_1. Now, ready time of resource R_1 is 20. Note that, task T_2 has 94 time units in resource R_1. So, task T_2 is assigned to resource R_3. Similarly, task T_3 and task T_4 are assigned to resource R_4 and resource R_2 respectively. The makespan is 81.

C. *Opportunistic Load Balancing* (*OLB*) *Heuristic*

It assigns the task to the resource which is idle soonest. It is not considering the execution time and/or the completion time of the task. If two resources are idle in same time for a scenario, then one of the resources is selected randomly. It only considers the resource ready time. Each task takes $O(r)$ time to find a resource [9, 12].

TABLE II. A 5 × 3 Expected ET Matrix

Task / Resource	R_1	R_2	R_3
RT	89	73	50
T_1	70	80	108
T_2	285	80	301
T_3	189	290	76
T_4	60	97	115
T_5	135	367	37

Let us consider an example shown in Table II. As per RT, resource R_3 is idle in 50. So, task T_1 is assigned to resource R_3 without considering the execution time. Next task T_2 is assigned to resource R_2 because it is idle soonest. Similarly, task T_3, task T_4 and task T_5 are assigned to resource R_1, resource R_2 and resource R_3 respectively. The makespan is 278.

D. *K-Percent Best* (*KPB*) *Heuristic*

It selects the resource based upon the value of K. When the value of K is 100, it chooses one resource from all available resources. For the same value of K, it works like MCT heuristic. For $K = (100/r)$, it works like MET heuristic. If K value is small then it selects one resource from very few resource(s). If K value is close to 100, then it selects one resource from more resources. The value of K makes a subset of resources. If K value is close to 50 then it selects best 50% resources. It also reduces time of a task to map with all resources. But it may lead to load imbalance problem. KPB has an overall complexity of $O(r \log r)$ time [12].

Let us consider the same example shown in Table I. Assume that the value of K is 50. So, it will select one resource from best two resources because we have four resources in the Table I matrix. The best two resources for task T_1 are resource R_1 and resource R_3. Then it uses MCT heuristic to assign a task to the resource. So, task T_1 is assigned to resource R_1. Like task T_1, the best two resources for task T_2 are resource R_1 and resource R_3. But it gives less completion time in resource R_3. So, it is assigned to resource R_3. Similarly, task T_3 and task T_4 are assigned to resource R_4 and resource R_1 respectively. The makespan is 81.

E. Switching Algorithm (SA)

It is a hybrid algorithm. It uses both MET and MCT heuristic. Let r_{max} indicates the maximum ready time of all available machines, r_{min} indicates the minimum ready time of all available machines and Π indicates the load balance index. Π can be calculated using Equation 2. The value of Π lies in between 0 to 1. In SA, two threshold values are used i.e. Π_l (low load balance index) and Π_h (high load balance index). The initial value of Π is set to zero and it starts with MCT heuristic. If Π is reached to Π_h or above, then it uses MET heuristic to decrease Π. If Π is decreased to Π_l or below, then it uses MCT heuristic to increase Π. SA has an overall complexity of $O(r)$ time [12].

$$\Pi = r_{min} / r_{max} \tag{2}$$

Let us consider an example shown in Table II. Assume that the value of Π_l = 0.2 and Π_h = 0.6 respectively. Initially, Π value is (50/89) = 0.562 and MCT heuristic is applied to the first task. Task T_1 is assigned to resource R_2. Then r_{min} and r_{max} are calculated. Here r_{min} is 50 and r_{max} is 153. Π = (50/153) = 0.326. So, we apply MCT heuristic for the upcoming task. Task T_2 is assigned to resource R_2. The value of r_{min} is updated to 50 and r_{max} is updated to 233. Π = (50/233) = 0.214. Again, we apply MCT heuristic for the upcoming task. Task T_3 is assigned to resource R_3. The value of r_{min} is updated to 89 and r_{max} is updated to 233. Π = (89/233) = 0.381. Task T_4 and task T_5 are assigned to resource R_1 and resource R_3 respectively. Finally Π is 0.63. So, the upcoming task is executed using MET heuristic. The makespan is 233.

IV. PROPOSED HEURISTICS

Heuristics are categorized into two modes: immediate (online) mode and batch (offline) mode. A task can be mapped to a resource as quickly as possible. It is referred as immediate mode heuristic. In contrary, a group of tasks can be mapped to the resources; it is referred as batch mode heuristic. We propose three immediate mode heuristics: (i) First-DualMake (F-DM) (ii) Best-DualMake (B-DM) and (iii) Worst-DualMake (W-DM). These heuristics are based on the idle time of the resource. The goal of these heuristic is to reduce the idle time of each resource instead of task completion time. For this, we need to calculate the resource ready time. It is called as Pre-makespan. It is the maximum ready time of all available resources. After each task has been executed, the makespan is recalculated. It is called as Post-makespan. In each heuristic, the name DualMake stands for Pre-makespan and Post-makespan.

A. Notations Used and Their Definitions

Notation	Definition
R_j	j^{th} Resource
T_i	i^{th} Task
RT_j	Initial Ready Time of resource j
M_{pr}	Pre-Makespan
M_{po}	Post-Makespan
$T_I[R_j]$	Idle Time of Resource j
$RT\ [R_j]$	Ready Time of Resource j
ET_{ij}	Execution Time of task i on Resource j
$RIT\ [R_j]$	Remaining Idle Time of resource j
R_c	Current Resource
X	Total number of tasks
Y	Total number of resources

B. First-DualMake (F-DM) Heuristic

In this heuristic, the first indicates the first available resource for the upcoming task. If there is no resource available, then the first task is assigned to most probable idle time resource. It first searches for available resources which has the enough idle time. It stops when it finds an available resource. Makespan is calculated after each assignment of task. We present the pseudo code of our proposed F-DM heuristic as follows.

```
1.  for all resource Rj
2.      Read RTj
3.  end for
4.  Calculate the Mpr
5.  for all resource Rj
6.      TI [Rj] = Mpr – RT [Rj]
7.  end for
8.  Sort the resource Rj with respect to TI [Rj] in
    ascending order
9.  do Ti ∈ T
10.     for all resource Rj
11.         RIT [Rj] = TI [Rj] - ETij
12.         Rc = Rj
13.         if RIT [Rj] ≥ 0
14.             Go to Step 18
15.         end if
16.     end for
17. end do
18. Assign Ti to resource Rc
19. Calculate the Mpo
```

First the ready time of each resource is determined. For loop in lines 1 to 3 of the heuristic calculates the ready time. In line 4, Pre-makespan is calculated. To determine the idle time of each resource, we need to calculate the difference between Pre-makespan and ready time of the resource. Line 5 to 7 for loop shows how to calculate the idle time. Then line 8 gives the ascending order of the idle time. Resources are sorted accordingly. Task assignment is done in line 9 to 18. The do loop is used to select one resource which is suitable for the task i. In line 11, the remaining idle time is calculated. It is the difference between idle time and the execution time of the task i on the resource j. It may be a negative value. If all values are negative, then it is assigned to most probable idle time resource. Line 12 shows this one. For positive values, the task T_i is assigned to resource R_c. Finally, we calculate the Post-makespan. The heuristic shown above is applicable for one task. For multiple tasks, we need to iterate the heuristic multiple times. Let us consider an example shown in Table III. There are four tasks <T_1, T_2, T_3 and T_4> and two resources <R_1 and R_2>. RT in Table III shows the ready time of the resources. Assume that task is arriving one after another in numeric order. In F-DM heuristic, Pre-makespan is calculated. It is 112 in this example.

TABLE III. A 4 × 2 EXPECTED ET MATRIX

Task / Resource	R_1	R_2
RT	85	112
T_1	41	46
T_2	7	11
T_3	32	14
T_4	24	28

Idle time of resource R_1 and resource R_2 is 27 and 0 respectively. Then the idle times are sorted in ascending order. It is 0 and 27 respectively. Accordingly, resources are sorted i.e. resource R_2 and resource R_1. Task T_1 takes 41 and 46 in resource R_1 and resource R_2 respectively. No resource is capable to execute task T_1. So, it is forcibly assigned to resource R_1 because it has more idle time. The Post-makespan is 126. The idle time of resource R_1 and resource R_2 is updated to 0 and 14 respectively. Task T_2 takes 7 and 11 in resource R_1 and resource R_2 respectively. The task T_2 can be assigned to resource R_2 because the difference between the idle time of resource R_2 and execution time of task T_2 on resource R_2 is greater than and equal to zero. Now, the Post-makespan is same as the previous iteration. Similarly, task T_3 and task T_4 are assigned to resource R_2 and resource R_1 respectively. The makespan is 150.

C. Best-DualMake (B-DM) Heuristic

In this heuristic, the best indicates the best available resource for the upcoming task. If no resource is available to map, then the task is assigned to most probable idle time resource. It searches the entire available resources and chooses a resource which is the smallest idle time. Unlike the F-DM, this heuristic is an alternative to choose one resource. It works like the F-DM if no resource has sufficient idle time. We present the pseudo code of our proposed B-DM heuristic as follows.

```
1.  for all resource R_j
2.      Read RT_j
3.  end for
4.  Calculate M_pr
5.  for all resource R_j
6.      T_I [R_j] = M_pr – RT [R_j]
7.  end for
8.  do T_i ∈ T
9.      for all resource R_j
10.         RIT [R_j] = T_I [R_j] – ET_ij
11.     end for
12.     Sort RIT [R_j] and its corresponding R_j in
    ascending order
13.     do RIT [R_j] < 0 && j < Y
14.         j = j + 1.
15.     end do
16. end do
17. Assign T_i to resource in index j
18. Calculate the M_po
```

Like F-DM, it calculates the ready time of each resource. Line 1 to 7 in B-DM is same as F-DM. In B-DM, the resources are not sorted before assignment. Remaining idle time is calculated in line 10. Tasks are sorted using remaining idle time. Thereafter, resources are sorted accordingly. The do loop in lines 13 to 15 do loop finds an idle resource for task T_i. It iterates until the remaining idle time value is positive as well as the j value is less than the number of resources. If the iteration fails, then it is assigned to the most probable idle resource. Finally, the Post-makespan is calculated. It is the Pre-makespan for the next task. Let us consider an example shown in Table IV. There are four tasks <T_1, T_2, T_3 and T_4> and three resources <R_1, R_2 and R_3>. RT in Table IV shows the ready time of the resources. At first, Pre-makespan is calculated. It is 112 in the Table IV example. Idle time of resource R_1, resource R_2 and resource R_3 are 27, 0 and 17 respectively. Task T_1 takes 41, 46 and 43 in resource R_1, resource R_2 and resource R_3 respectively. Then it calculates remaining idle time. The remaining idle time is sorted in ascending order. The values are negative. It means no resource is able to execute task T_1. So, it is forcibly assigned to resource R_1 as it has more idle time. The Post-makespan is 126. The idle time of resource R_1, resource R_2 and resource R_3 are updated to 0, 14 and 31 respectively. Task T_2 takes 7, 11 and 14 in resource R_1, resource R_2 and resource R_3 respectively. The task T_2 can be assigned to either resource R_2 or resource R_3. According to B-DM heuristic, it is assigned to resource R_2 because the idle time of resource R_2 is reduced to an extent. Now, the Post-makespan is same as the previous iteration. Similarly, task T_3 and task T_4 are assigned to resource R_3.The makespan is 126. Generally, both B-DM and W-DM heuristics are not measured in two resource environment because it acts like the F-DM heuristic.

TABLE IV. A 4 × 3 EXPECTED ET MATRIX

Task / Resource	R_1	R_2	R_3
RT	85	112	95
T_1	41	46	43
T_2	7	11	14
T_3	32	14	23
T_4	24	28	6

D. Worst-DualMake (W-DM) Heuristic

In this heuristic, the worst indicates the worst available resource for the upcoming task. It is similar to B-DM. But, it selects the worst resource instead of best resource. It searches the entire available resources and chooses a resource which is the largest idle time. It assigns the task to the resource which holds the largest idle time. It is same as MCT heuristic. The idle time of a task in W-DM same as the earliest completion time of a task. As a part of the complete idle time scenario, we have shown in this paper. We present the pseudo code of our proposed W-DM heuristic as follows.

```
1.  for all resource R_j
2.      Read RT_j
3.  end for
4.  Calculate M_pr
5.  for all resource R_j
6.      T_I [R_j] = M_pr – RT [R_j]
7.  end for
8.  do T_i ∈ T
9.      for all resource R_j
10.         RIT [R_j] = T_I [R_j] - ET_ij
11.     end for
12.     Find max(RIT [R_j])
13. end do
14. Assign T_i to resource R_j
15. Calculate the M_po
```

In W-DM, Line 1 to 11 is same as B-DM. In this heuristic, Line 12 finds the maximum remaining idle time of entire resources. If none of the resource is sufficient idle time, then it works like the F-DM heuristic. Unlike the F-DM and B-DM heuristic, it does not require sorting function. Let us consider an example shown in Table IV. There are four tasks <T_1, T_2, T_3 and T_4> and three resources <R_1, R_2 and R_3>. At first, Pre-makespan is calculated. It is 112 in the Table IV example. Idle time of resource R_1, resource R_2 and resource R_3 are 27, 0 and 17 respectively. Task T_1 takes 41, 46 and 23 in resource R_1, resource R_2 and resource R_3 respectively. Then it calculates remaining idle time. The calculated values are negative. It means no resource is able to execute task T_1. So, it is forcibly assigned to resource R_1 because it has more idle time. The Post-makespan is 126. The idle time of resource R_1, resource R_2 and resource R_3 are updated to 0, 14 and 31 respectively. Task T_2 takes 7, 11 and 14 in resource R_1, resource R_2 and resource R_3 respectively. The task T_2 can be assigned to either resource R_2 or resource R_3. According to W-DM heuristic, it is assigned to resource R_3 because the difference between the idle time of resource R_3 and the execution time of task T_2 on resource R_3 is more than the idle time of resource R_2 and the execution time of task T_2. Now, the Post-makespan is same as the previous iteration. Similarly, task T_3 and task T_4 are assigned to resource R_2 and resource R_3 respectively. The makespan is 126.

V. SIMULATION AND RESULTS

The proposed heuristics are implemented and compared using the benchmark instances by Braun et al. [10]. We have used MATLAB R2010b to simulate the heuristics. We have taken different sizes of EET matrices such as 512 tasks and 16 resources and 1024 tasks and 32 resources. In each size, 12 different types of instances are compared. The instances are consisting of three parameters: distribution, the nature of the matrix and task-resource heterogeneity. The general representations of these instances are u_a_bbcc.o. Here, 'u' indicate the distribution is uniform followed by 'a' indicates the nature of the matrix i.e. c – consistent, i – inconsistent and s – semi-consistent. Then, bb indicates the task heterogeneity and cc indicates the resource heterogeneity i.e. either high (hi) or low (lo). We have taken Π_l = 0.6 and Π_h = 0.9 in SA heuristic for all types of instance. We have used two performance measures to evaluate the heuristics. They are makespan and resource utilization. First, we simulated for 512 tasks and 16 resources. The existing heuristic results for 512 × 16 instances are also shown in Xhafa et al. and Chaturvedi et al. [11, 14]. The makespan comparison of the proposed and the existing heuristics is shown in Table V. Next, we simulated for 1024 tasks and 32 resources. The makespan comparison of the proposed and the existing heuristics is shown in Table VI. The resource utilization comparisons of both data sets are jointly shown in Table VII. In this paper, we have seen that W-DM (or MCT) heuristic and B-DM heuristic is best among all the heuristics present in the literature (for consistent matrices). The KPB heuristic is best for inconsistent matrices among all other heuristics.

VI. CONCLUSION

In this paper, eight immediate mode heuristics are discussed and implemented in MATLAB R2010b. None of the heuristic is giving better result in all instances. As we know, scheduling in heterogeneous grid environment is a NP-Complete problem; there is no such algorithm exist that will solve in a polynomial time. We have observed that MCT heuristic gives better results in consistent. But, KPB gives better results in 512 × 16 inconsistent matrices (hilo, lohi and lolo instances) and semi-consistent matrices (hihi, hilo and lolo instances). Among the three proposed heuristics, the W-DM heuristic is similar to MCT heuristic. After MCT heuristic, B-DM heuristic gives better results in consistent matrices.

REFERENCES

[1] H. Attiya and J. Welch, "Distributed Computing", John Wiley and Sons, 2000.

[2] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments", 9th Heterogeneous Computing Workshop, pp. 349-363, 2000.

[3] A. Chakrabarti, "Grid Computing Security", Springer, 2007.

[4] I. Foster and C. Kesselman, "The Grid – Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, 1998.

[5] R. Buyya, "High Performance Cluster Computing", Pearson Education, 2008.

[6] 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.

[7] 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.

[8] 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.

[9] R. Armstrong, D. Hensgen and T. Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions", Seventh IEEE Heterogenous Computing Workshop, pp. 79-87, 1998.

[10] 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.

[11] 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.

[12] 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.

[13] D. D. H. Miriam and K. S. Easwarakumar, "A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems", International Journal of Computer Science Issues, Vol. 7, No. 4, pp. 8-18, 2010.

[14] F. Xhafa, J. Carretero, L. Barolli and A. Durresi, "Immediate mode scheduling in grid systems", International Journal of Web and Grid Services, Vol. 3, No. 2, pp. 219-236, 2007.

[15] A. Rasooli, M. Mirza-Aghatabar and S. Khorsandi, "Introduction of Novel Rule Based Algorithms for Scheduling in Grid Computing Systems", Second Asia International Conference on Modelling & Simulation, 2008.

[16] C. Grosan, "A Multiobjective Metaheuristic for Job-Shop Scheduling", Informatica, Vol. 54, No. 1, pp. 97-102, 2009.

[17] H. Dail, F. Berman, and H. Casanova, "A Decoupled Scheduling Approach for Grid Application Development Environments", Journal of Parallel and Distributed Computing, Vol. 63, No. 5, pp. 505-524, 2003.

[18] S. H. Chin, J. H. Lee, T. M. Yoon and H. C. Yu, "List Scheduling Method for Srevice Oriented Grid Applications", Second International Conference on Semantics, Knowledge and Grid, 2006.

[19] 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.

[20] F. Dong, J. Luo, L. Gao and L. Ge, "A Grid Task Scheduling Algorithm Based on QoS Priority Grouping", Fifth International Conference on Grid and Cooperative Computing, 2006.

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

[22] H. Decai, Y. Yuan, Z. Li-jun and Z. Ke-qin, "Research on Tasks Scheduling Algorithms for Dynamic and Uncertain Computing Grid Based on a+bi Connection Number of SPA", Journal of Software, Vol. 4, No. 10, pp. 1102-1109, 2009.

[23] S. Parsa and R. Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications, Vol. 3, No. 4, pp. 91-99, 2009.

[24] 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.

[25] S. K. Panda, S. K. Bhoi and P. M. Khilar, "RRTS: A Task Scheduling Algorithm to Minimize Makespan in Grid Environment", Second International Conference on Internet Computing and Information Communications, Springer, Vol. 216, pp. 2012.

[26] S. K. Panda and P. M. Khilar, "A Three-Stage Approach for Grid Task Scheduling", Second IEEE International Conference on Parallel, Distributed and Grid Computing, pp. 441-446, 2012.

[27] 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.

[28] Z. Jinquan, N. Lina and J. Changjun, "A Heuristic Scheduling Strategy for Independent Tasks on Grid", Eighth International Conference on High-Performance Computing in Asia-Pacific Region, 2005.

TABLE V. NUMERICAL RESULTS OF MAKESPAN VALUE FOR 512 × 16 INSTANCES

Instance	MET	MCT	OLB	KPB	SA	F-DM	B-DM	W-DM
u_c_hihi	4.7472E+07	1.1423E+07	1.4377E+07	1.2497E+07	1.2613E+07	1.3359E+07	1.1980E+07	1.1423E+07
u_c_hilo	1.1851E+06	1.8589E+05	2.2105E+05	2.0115E+05	1.9455E+05	1.9837E+05	1.9480E+05	1.8589E+05
u_c_lohi	1.4531E+06	3.7830E+05	4.7736E+05	4.0029E+05	4.2627E+05	4.4870E+05	3.9477E+05	3.7830E+05
u_c_lolo	3.9582E+04	6.3601E+03	7.3066E+03	6.8463E+03	8.1671E+03	6.7718E+03	6.4801E+03	6.3601E+03
u_i_hihi	4.5085E+06	4.4136E+06	2.6102E+07	4.5087E+06	4.6922E+06	1.0856E+07	7.6376E+06	4.4136E+06
u_i_hilo	9.6610E+04	9.4856E+04	2.7279E+05	9.3006E+04	1.0298E+05	1.8464E+05	1.3080E+05	9.4856E+04
u_i_lohi	1.8569E+05	1.4382E+05	8.3361E+05	1.4382E+05	1.4391E+05	3.3721E+05	2.5974E+05	1.4382E+05
u_i_lolo	3.3993E+03	3.1374E+03	8.9380E+03	3.1230E+03	3.4853E+03	5.4797E+03	4.4006E+03	3.1374E+03
u_s_hihi	2.5162E+07	6.6939E+06	1.9465E+07	6.5142E+06	7.1277E+06	1.2331E+07	9.3984E+06	6.6939E+06
u_s_hilo	6.0536E+05	1.2659E+05	2.5036E+05	1.2354E+05	1.4905E+05	1.6985E+05	1.5567E+05	1.2659E+05
u_s_lohi	6.7469E+05	1.8615E+05	6.0323E+05	1.8796E+05	1.9432E+05	3.6149E+05	2.8610E+05	1.8615E+05
u_s_lolo	2.1042E+04	4.4361E+03	8.9384E+03	4.4052E+03	5.8370E+03	6.2136E+03	5.5996E+03	4.4361E+03

TABLE VI. NUMERICAL RESULTS OF MAKESPAN VALUE FOR 1024 × 32 INSTANCES

Instance	MET	MCT	OLB	SA	F-DM	B-DM	W-DM
u_c_hihi	1.5447E+08	3.2833E+07	4.2817E+07	3.7301E+07	3.5554E+07	3.3651E+07	3.2833E+07
u_c_hilo	1.5504E+07	3.2458E+06	4.4054E+06	3.2458E+06	3.9253E+06	3.9253E+06	3.2458E+06
u_c_lohi	1.4151E+04	3.0587E+03	4.4132E+03	3.0587E+03	3.7370E+03	3.2375E+03	3.0587E+03
u_c_lolo	1.5675E+03	3.2628E+02	4.4475E+02	4.1438E+02	4.0760E+02	3.3570E+02	3.2628E+02
u_i_hihi	7.4620E+06	7.5671E+06	8.4914E+07	7.5671E+06	2.3937E+07	1.6626E+07	7.5671E+06
u_i_hilo	7.6598E+05	7.1313E+05	7.8322E+06	7.1313E+05	2.7569E+06	1.5649E+06	7.1313E+05
u_i_lohi	8.5439E+02	7.5410E+02	8.6143E+03	7.5410E+02	2.2689E+03	1.7735E+03	7.5410E+02
u_i_lolo	9.1120E+01	7.2390E+01	9.0081E+02	7.2390E+01	2.8137E+02	1.5366E+02	7.2390E+01
u_s_hihi	8.4821E+07	1.9008E+07	7.7562E+07	1.9008E+07	3.5030E+07	2.6025E+07	1.9008E+07
u_s_hilo	8.0988E+06	1.8255E+06	8.1962E+06	1.8255E+06	3.7373E+06	2.4675E+06	1.8255E+06
u_s_lohi	8.3377E+03	1.8220E+03	7.9978E+03	1.8220E+03	4.3091E+03	2.4676E+03	1.8220E+03
u_s_lolo	8.0161E+02	1.9423E+02	8.2890E+02	1.9423E+02	3.6628E+02	2.6774E+02	1.9423E+02

TABLE VII. NUMERICAL RESULTS OF RESOURCE UTILISATION VALUE FOR BOTH 512 × 16 AND 1024 × 32 INSTANCES

Instance	MET (512 × 16)	MCT (512 × 16)	OLB (512 × 16)	KPB (512 × 16)	SA (512 × 16)	F-DM (512 × 16)	B-DM (512 × 16)	W-DM (512 × 16)	MET (1024 × 32)	MCT (1024 × 32)	OLB (1024 × 32)	SA (1024 × 32)	F-DM (1024 × 32)	B-DM (1024 × 32)	W-DM (1024 × 32)
u_c_hihi	1	0.9539	0.9467	0.972	0.8905	0.9173	0.9740	0.9539	1	0.9355	0.8980	0.8058	0.9702	0.9698	0.9355
u_c_hilo	1	0.9707	0.9203	0.974	0.9209	0.9681	0.9736	0.9707	1	0.9461	0.8886	0.9461	0.8625	0.9562	0.9461
u_c_lohi	1	0.9690	0.9285	0.969	0.8326	0.9206	0.9762	0.9690	1	0.9226	0.8625	0.9226	0.8623	0.9424	0.9226
u_c_lolo	1	0.9515	0.9232	0.960	0.7279	0.9566	0.9733	0.9515	1	0.9501	0.8646	0.7075	0.8666	0.9713	0.9501
u_i_hihi	0.6286	0.9329	0.9512	0.929	0.8469	0.9546	0.9801	0.9329	0.6605	0.9122	0.9410	0.9122	0.9549	0.9665	0.9122
u_i_hilo	0.7506	0.9598	0.9559	0.954	0.8167	0.9196	0.9840	0.9598	0.6058	0.9126	0.9621	0.9126	0.9726	0.9607	0.9126
u_i_lohi	0.5366	0.9496	0.9340	0.937	0.9481	0.9547	0.9604	0.9496	0.5799	0.9167	0.9613	0.9167	0.9741	0.9733	0.9167
u_i_lolo	0.7404	0.9657	0.9796	0.968	0.7977	0.9674	0.9782	0.9657	0.5264	0.9178	0.9302	0.9178	0.9885	0.9768	0.9178
u_s_hihi	0.1971	0.9283	0.9671	0.928	0.8631	0.9863	0.9670	0.9283	0.0577	0.9134	0.9290	0.9134	0.9594	0.9711	0.9134
u_s_hilo	0.2142	0.9383	0.9246	0.951	0.7813	0.9802	0.9923	0.9383	0.0602	0.9326	0.9510	0.9326	0.9771	0.9746	0.9326
u_s_lohi	0.2167	0.9539	0.9620	0.946	0.8911	0.9831	0.9813	0.9539	0.0604	0.8980	0.9430	0.8980	0.8548	0.9705	0.8980
u_s_lolo	0.2212	0.9519	0.9510	0.950	0.7086	0.9514	0.9758	0.9519	0.0614	0.9037	0.9489	0.9037	0.8977	0.9711	0.9037