An Efficient Batch-Mode Scheduling Heuristic Based on Load Balancing

Jyoti bansal[1], Vishu Narula [2], Dr. Shaveta [3], Dr. Paramjit Singh[4]
[1] Research Scholar, PTU, kapurthala, India
[1] erjyoti.2009@rediffimail.com
[2]BFCET, Bathinda, India
[2] vishunarula90@gmail.com
[3,4] GZS, PTU Campus, Bathinda, India
[3]garg_shavy@yahoo.com
[4]param2009@yahoo.com

Abstract— In this paper, an efficient Batch-Mode scheduling heuristics have been proposed for balancing the load in the Desktop Grid environment. The proposed heuristic works in two phases: In first phase, we are making a schedule of Expected Execution Time (EET_{uv}) for all tasks w.r.t. resources by following Max-Min for m tasks & Min-Min and Max-Min alternatively for remaining n-m tasks, where n & m are the number of tasks (T_u) and resources (R_v) respectively. Then scheduling is being done as per the minimum EET_{uv} taken by the tasks w.r.t. the resources without considering the load imbalance on resources. In second phase, to remove the load imbalance, tasks will get transferred from maximally loaded to minimally loaded resources. The concept has tested experimentally by using GridSim 5.2, and results proves that proposed heuristic performs well on comparing with Min-Min, Max-Min and LJFR-SJFR heuristic for minimizing Makespan, Flowtime and Average Completion Time value.

Keywords- Desktop Grid computing, Min-Min, Max-Min & LJFR-SJFR.

I. INTRODUCTION

The enormous reputation of Internet has made another much expansive scale open door for Grid computing. Points of fact, many desktop PCs, whose idle cycles can be changed to run Grid applications, are joined with wide-zone systems both in the business enterprises and in the home. These new stages for high throughput applications are called Desktop Grids [1,2]. Desktop grid environment faces new challenges day by day due to its heterogeneous nature [3]. So to manage the resources in desktop grid environment we require some scheduling heuristic and these scheduling heuristic in desktop grid is separated into two categories i.e., knowledge-based [4,5] and knowledge-free scheduling [6]. Further these scheduling heuristics are subdivided into online and batch-mode scheduling [7]. In online scheduling, the scheduler will assign the task to the machine as soon as the task will come to the scheduler and this decision of scheduler will not get changed once it has been computed. Conversely, in the batch mode, tasks are not directly mapped onto the machines as they arrive, instead they are collected into a set that is examined for mapping at prescheduled times which is called mapping events [8,9].

The rest of this paper is organized as: Section II will characterize related work. Section III will provide the heuristics description. Section IV will analyse the test problem. Section V will describe the proposed heuristic. Section VI will define the comparison and experimental results. Section VII will conclude the work and provides the future work.

II. LITRATURE SURVEY

Scheduling is a key issue in grid and desktop grid computing. In order to improve system performance we need to schedule tasks to run at the right place at the right time. In particular, when task compete for resources that are not preemptive, it is very important to interleave the usage of resources so that system utilization and quality of service are maintained at the same time. Min-Min, Max-Min, LJFR-SJFR and many more heuristics have being used for task scheduling in grid environment. The research on them received good results and their efficiency had been proved. There are also many related improved work under study.

Like in [9], to achieve the high throughput computing in a Grid environment authors' proposed an adaptive scheduling heuristic by adding guided QoS component into the conventional Min-Min heuristic to form the QoS guided Min-Min heuristic. Furthermore in [27], to provide the matching application requirements with available resources authors presented a job scheduling heuristic again based on quality of service (QoS). In [10], the two batch-mode heuristics i.e., Min-Min and Max-Min has been discussed. The authors' have discussed its disadvantages and have tried to overcome it. In [11], authors' presented new heuristic 'RASA'. This heuristic overcomes the disadvantages of traditional Min-Min and Max-Min heuristics. In [12] & [13], authors' proposed a new heuristic to minimize the Makespan, Flowtime and maximizes resources utilization. In [14] & [15], the exploratory results will pronounce that Min-Min heuristic will give best result for minimizing Flowtime and the

proposed heuristic (Min-Max) will announce the best results for minimizing Makespan. After analyzing Min-Min, Max-Min and LJFR-SJFR in [58], authors' presented a Differential Evolution (DE) for scheduling in grid computing environments to deliver improved Makespan. In [16,17,18], authors' proposed new heuristic for balancing the resource load by adding new features in Min-Min and Max-Min.

III. HEURISTIC DETAILS.

This section provides the description of three batch-mode scheduling heuristics and some parameters for mapping tasks to available resources in HC environments.

A. Min-Min Heuristic

This section provides the description of three batch-mode scheduling heuristics and some parameters for mapping tasks to available resources in HC environments. Min-Min algorithm begins by collecting all the Meta Tasks (MT) of all unassigned tasks and it will calculate the Completion Time (CT) for all the tasks. The Min-Min algorithm works in two phases. So in first phase, it will get the minimum expected completion time for each task in MT:

$$M = \{\min[\text{completion\_time}(T_u, R_v)] \text{ for} (1 \le u \le n, 1 \le v \le m)\}$$

In Second phase, it will select the task with overall minimum expected completion time from MT and will assign it to the corresponding resource. At the end, it will remove the task from MT and repeat process until all tasks in the MT are mapped [21].

B. Max-Min Heuristic

Max-Min is very similar to Min-Min except phase 2. In first phase, it begins with the set *U* of all unmapped tasks. Then, the set of minimum completion times will get calculated same as Min-Min. In second phase, it assigns the tasks with maximum expected completion time to the resource and the workload of the selected resource will get updated. At the end, the newly mapped task will get removed from *U* and the process will repeat until all tasks will get mapped [21].

C. LJFR-SJFR Heuristic

Longest Job to Fastest Resource-Shortest Job to Fastest Resource (LJFR-SJFR) heuristic starts by selecting every single unmapped task. Here, LJFR-SJFR heuristic will also works in two stages. In first stage, it will first allot the biggest task to the resources by utilizing Max-min heuristic i.e., it will first choose those tasks that have greatest expected completion time. In second stage, remaining tasks will get allocated by utilizing min-min and max-min heuristic alternatively i.e. smallest tasks on quickest resource took after by biggest task on speediest resource [16]. Longest Task to Fastest Resource-Shortest Task to Fastest Resource (LJFR-SJFR) heuristics assigns longest task to fastest resource to minimize the makespan and assigns smallest task to fastest resource to minimize the flow time value. In the first phase: M numbers of tasks are allocated on m number of resources by using Max-Min heuristic as illustrated above. In second phase: remaining n-m tasks are allocated by using Min-Min and Max-Min heuristic alternatively i.e. smallest task on fastest resource followed by longest task on fastest resource. Process is followed until all unassigned tasks get mapped [14].

D. Makespan

It is the time difference between the start time of first task and finish time of the last task [22]. Basically Makespan measures the throughput of the system and it is defined as:

$$\text{Makespan} = \max\left(C_{u,u=1,\cdots,n}\right)$$

E. Flowtime

It is the sum of the finishing times of tasks [22]. It measures the quality of service of grid system and it is defined as:

$$\text{Flowtime} = \left(\sum C_{u,u=1,\cdots,n}\right)$$

F. Average Completion Time

It will return the average period of time taken by tasks to complete its execution and it is defined as:

$$\text{Average Completion Time} = \frac{\left(\sum C_{u,u=1,\ldots,n}\right)}{n}$$

IV. TEST PROBLEMS.

For reasonable comparison of distinctive scheduling heuristic, this segment will outline the illustrations of above characterized heuristics i.e., Min-Min, Max-Min & LJFR-SJFR. Assume that m resources $R_{v(v=1\ldots m)}$ need to process n tasks $T_{u(u=1\ldots n)}$. It is also assumed that the length of each task and the information of all available resources are known beforehand. Presently in like manner to the compare the heuristics table I, table II, table III will characterize the Expected Execution Time of every assignment.

TABLE I: EET_{vu} of Min-Min

Tasks in MI	**Resource(speed) in MIPS**		
Tasks (size)	R1(200)	R2(300)	R3(400)
T1(2000)	10	6.66	5
T2(3000)	15	10	7.5
T3(4000)	20	13.33	10
T4(8000)	40	26.66	20
T5(9000)	45	30	22.5
T6(11000)	55	36.66	27.5
T7(12000)	60	40	30
T8(15000)	75	50	37.5
T9(17000)	85	56.66	42.5
T10(20000)	100	66.66	50

TABLE II: EET_{vu} of Max-Min

Tasks in MI	**Resource(speed) in MIPS**		
Tasks (size)	R1(200)	R2(300)	R3(400)
T10(20000)	100	66.66	50
T9(17000)	85	56.66	42.5
T8(15000)	75	50	37.5
T7(12000)	60	40	30
T6(11000)	55	36.66	27.5
T5(9000)	45	30	22.5
T4(8000)	40	26.66	20
T3(4000)	20	13.33	10
T2(3000)	15	10	7.5
T1(2000)	10	6.66	5

TABLE III: EET_{vu} of LJFR-SJFR

Tasks in MI	**Resource(speed) in MIPS**		
Tasks (size)	R1(200)	R2(300)	R3(400)
T10(20000)	100	66.66	50
T9(17000)	85	56.66	42.5
T8(15000)	75	50	37.5
T1(2000)	10	6.66	5
T7(12000)	60	40	30
T2(3000)	15	10	7.5
T6(11000)	55	36.66	27.5
T3(4000)	20	13.33	10
T5(9000)	45	30	22.5
T4(8000)	40	26.66	20

By experimentally performing the corresponding succession of steps relating to every heuristic by using GridSim 5.2 [23], the table IV, table V & table VI will mirror the outcomes of tasks T_u on resources R_v individually.

TABLE IV: Result of Min-Min

Resources	**Speed**	**Task Assigned**
R1	200	T6,T10
R2	300	T2,T5,T8
R3	400	T1,T3,T4,T7,T9

TABLE V: Result of Max-Min

Resources	Speed	Task Assigned
R1	200	T3,T7
R2	300	T2,T5,T8,T10
R3	400	T1,T4,T6,T9

TABLE VI: Result of LJFR-SJFR

Resources	Speed	Task Assigned
R1	200	T8,T3
R2	300	T7,T2,T6
R3	400	T10,T9,T1,T4,T5

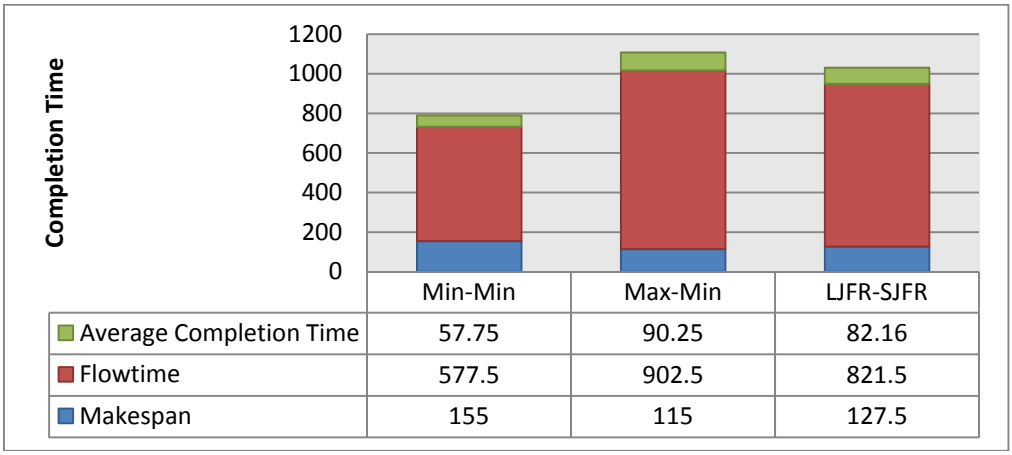

Figure I: Comparison of Three Scheduling Heuristics

V. PROPOSED HEURISTIC

As it is evident from the figure I, Min-Min heuristic can minimize Flowtime and Average Completion time value better than others. Conversely Max-Min provides the better Makespan value than the others. Furthermore LJFR-SJFR can minimize the Flowtime and Average Completion time value better than the Max-Min. Separated this we have investigated that: In Min-Min, the schedule won't stays optimal when the smaller number of tasks gets more than larger one & in Max-Min, the schedule won't stay optimal when the quantity of larger tasks gets more than smaller one. Now all these circumstances will leads to load imbalance on resources. So there is need to have some load balancing heuristic for dealing with this issue. To resolve this problem, this paper proposes the new heuristic which works in two phases: In first phase: We are making a schedule of Expected Execution Time (EET_{uv}) for all tasks w.r.t. resources by following Max-Min heuristic for m number of tasks & Min-Min and Max-Min alternatively for remaining n-m tasks.

$$\text{Where, EET} = \frac{\text{Task Length}}{\text{Resouce Speed}}$$

Now for scheduling, the tasks having minimum Expected Execution Time (EET_{uv}) will get chosen and allocated to the corresponding resource without considering the load imbalanced. In second phase: To remove the load imbalance, we will identify the maximally loaded resource and minimally loaded resource. Then for m number of tasks having minimum EET_{uv} value on maximally loaded resource are selected and transferred to the minimally loaded resource according to Minimum Completion Time of task w.r.t corresponding resource. The scheduling process based on proposed heuristic that is experimented in this paper is represented in figure II.

Phase 1:

1. For all tasks $T_{u(I\ to\ n)}$ in Q_b
2. For all resources $R_{v\ (I\ to\ m)}$
3. Compute the Expected Execution Time EET_{uv} corresponding to each Resources R_v
   EET_{uv} = Job length/resource speed
4. Do until all tasks in Q_b are mapped
5. Find the Resources that gives the earlier Expected Execution Time for each task(T_u)
6. If(n<=m)
7. For each Task in Q_b
8. Find the Task T_u with maximum earliest Expected Execution Time
9. Assign the task T_u to the corresponding Resource R_j
10. Update the Values of T_u from R_u & EET_{uv}
11. Else if (n>m)
12. For each task in Q_b find the task T_u with minimum earliest Expected Execution Time Q_b
13. Assign the task to the corresponds Resources
14. Update the values of T_u & R_v & Q_b
15. For each task in Q_b find the task J_k with maximum earlier Expected Execution Time
16. Assign the task T_k to the corresponding Resource R_j
17. Update the values of T_u & R_v & Q_b

Phase 2:

18. For all tasks $T_{u(I\ to\ n)}$ in Q_b
19. Find the maximally loaded Resource R_{max} minimally loaded Resource R_{min} & also find task T_{min} having minimum EET_{uv} & T_{max} having maximum EET_{uv}
20. Transfer the m Task T_{min} having minimum Expected Execution Time to the maximally loaded Resource R_{max} according to the minimum completion time on comparing resource.

Figure II: Scheduling process of Proposed Heuristic

VI. COMPARISON & EXPERIMENTAL RESULTS

To elaborate the concept of proposed technique, by using GridSim 5.2 [23] following example is considered with the same scenario of Expected Execution Time as illustrated in table 3, in which 10 tasks are computed on 3 resources. The result of phase 1 is illustrated in table VII, where all tasks get allocated to R3 (as it gives minimum EET_{uv} value) and this will lead to load imbalance. In second phase, to remove the load imbalance, we identify the maximally loaded resource and minimally loaded resource. As R3 will become maximally loaded resource, so from R3 we select 3 tasks with minimum Expected Execution Time value i.e., T1, T2 & T3 and transferred them onto the minimally loaded resource according to Minimum Completion Time of task w.r.t corresponding resource. After balancing the load the final result of proposed heuristic is illustrated in table VIII.

TABLE VII: Result of 1st Phase of Proposed heuristic

Resources	Speed	Task Assigned
R1	200	-
R2	300	-
R3	400	T10,T9,T8,T1,T7,T2,T6,T3,T5,T4

In second phase, to remove the load imbalance, we identify the maximally loaded resource and minimally loaded resource. As R3 will become maximally loaded resource, so from R3 we select 3 tasks with minimum Expected Execution Time value i.e., T1, T2 & T3 and transferred them onto the minimally loaded resource according to Minimum Completion Time of task w.r.t corresponding resource. After balancing the load the final result of proposed heuristic is illustrated in table VIII.

TABLE VIII: Result of proposed heuristic

Resources	Speed	Task Assigned
R1	200	T2
R2	300	T1, T3
R3	400	T10,T9,T8,T7,T6,T5,T4

With this we analyzed that, as we balance the load in between the resources the Makespan, Flowtime and Average Completion Time value will also get minimized. The obtained Makespan, Flowtime and Average Completion Time of respective mentioned heuristics are compared in figure III, IV, & V respectively. In these figures, the first column indicates the instance name, and the second, third, fourth and fifth column depicts the Makespan, Flowtime and Average Completion Time values of Min-Min, Max-Min, LJFR-SJFR and Proposed Heuristic. And this proves that Proposed Heuristic gives the best results for balancing the load in between the resources and also minimizes the Makespan, Flowtime and Average Completion Time value on comparing it with three batch-mode heuristics.

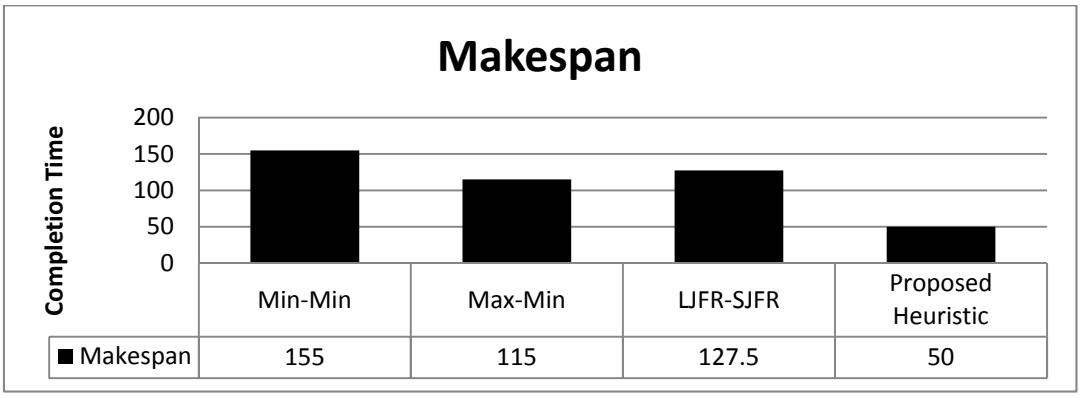

Figure III: Comparison results between heuristics on Makespan

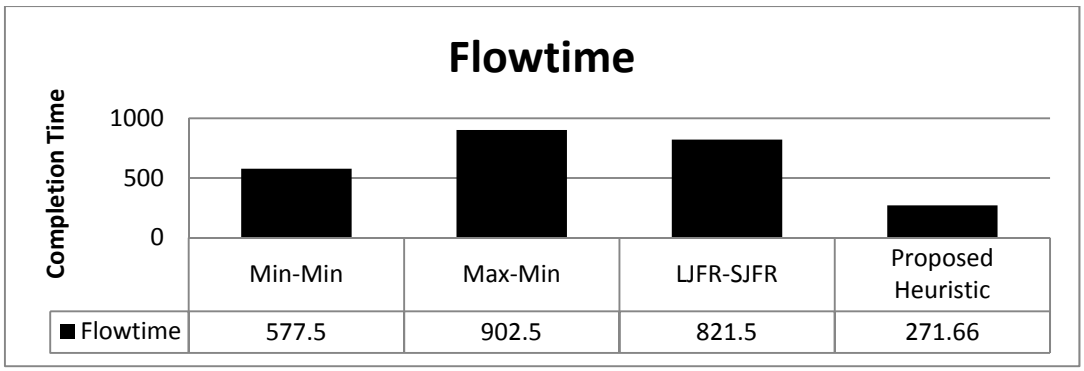

Figure IV: Comparison results between heuristics on Flowtime

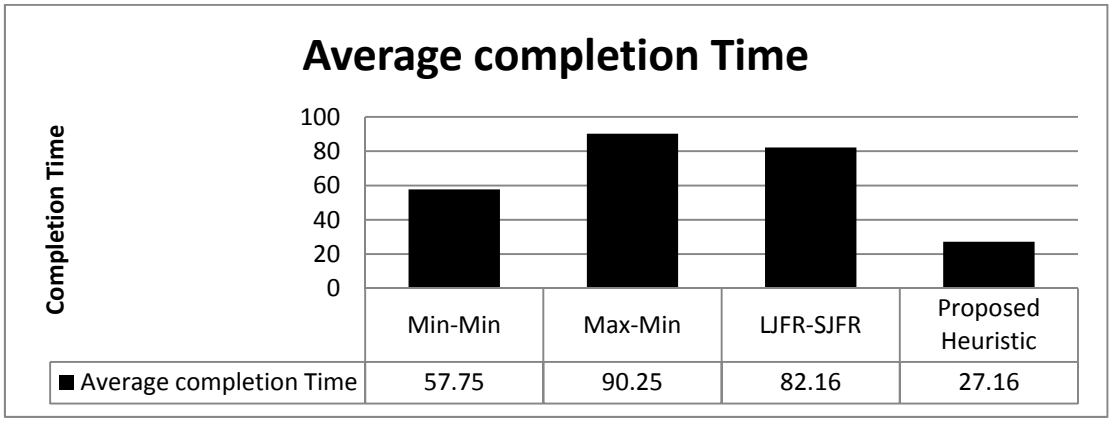

Figure V: Comparison results between heuristics on Average Completion Time

VII. CONCLUSION & FUTURE WORK

In this paper an efficient Batch-Mode Heuristic has proposed. The contribution of this paper is to provide load balancing heuristic which provides better Makespan, Flowtime and Average Completion Time value in the Desktop Grid environment. The computed results will prove that Proposed Heuristic performs well on comparing it with three traditional heuristics i.e., Min-Min, Max-Min and LJFR-SJFR. For future work, we have surveyed some of the essential factors that persuade the task scheduling i.e., Dependency among tasks, execution cost, communication cost, and error factors like node failure, data storage, network failure, task duplications, execution time length, and heterogenic network behavior, fault tolerance, security, dynamicity, performance, cost efficiency, Quality of Service, time dependency, resource recovery, resource allocation, and management, so on. All these factors play a very important role in desktop grid environment. So for future work we have decided to tackle with security issue and will try to provide secured desktop grid environment.

REFERENCES

[1] SungJin Choi, HongSoo Kim, et al. 2007, Characterizing and classifying desktop grid, In null, pp. 743-748, IEEE.

[2] Issam Al-Azzoni, Douglas G. Down, et al. 2010, Dynamic scheduling for heterogeneous desktop grids, Journal of Parallel and Distributed Computing', 70(12), pp.1231-1240, ELESEVIER.

[3] Choi, Sungjin, Rajkumar Buyya, et al. 2008, A taxonomy of desktop grids and its mapping to state of the art systems. Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Tech. Rep.

[4] Berman, Francine, Richard Wolski, et al. 2003, Adaptive computing on the grid using AppLeS. Parallel and Distributed Systems, IEEE Transactions on 14, no. 4, pp. 369-382.

[5] Casanova, Henri, Arnaud Legrand, et al. 2000, Heuristics for scheduling parameter sweep applications in grid environments. In Heterogeneous Computing Workshop, (HCW 2000) Proceedings. 9th, pp. 349-363, IEEE

[6] Da Silva, Daniel Paranhos, et al. 2003, Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In Euro-Par 2003 Parallel Processing, pp. 169-180. Springer Berlin Heidelberg.

[7] Muthucumaru Maheswaran, Shoukat Ali, et al. 1999, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In Heterogeneous Computing Workshop, (HCW'99) Proceedings Eighth, pp.30-44, IEEE.

[8] Braun, Tracy D., Howard Jay Siegel, et al. 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 61, no. 6, pp.810-837.

[9] Anglano, Cosimo, and Massimo Canonico. 2005, Fault-tolerant scheduling for bag-of-tasks grid applications. In Advances in Grid Computing-EGC 2005, pp. 630-639. Springer Berlin Heidelberg.

[10] He, XiaoShan, XianHe Sun, et al. 2003, QoS guided Min-Min heuristic for grid task scheduling. Journal of Computer Science and Technology 18, no. 4, pp.442-451.

[11] Wang, Yong, Shougeng Hu, and Gaifang Wang. 2009, A Strategy of Resource Scheduling for Grid Computing Based on QoS. In Information Science and Engineering (ICISE), 2009 1st International Conference on, pp. 82-85. IEEE.

[12] Izakian, Hesam, Ajith Abraham, et al. 2009, Performance comparison of six efficient pure heuristics for scheduling meta-tasks on heterogeneous distributed environments. Neural Network World 19, no. 6, pp.695-710.

[13] Chaturvedi, Anand K., et al. 2011, New heuristic for scheduling of independent tasks in computational grid. International Journal of Grid and Distributed Computing 4, no. 3, pp. 25-36.

[14] Parsa, Saeed, and Reza Entezari-Maleki. 2012, RASA: A new task scheduling algorithm in grid environment. World Applied sciences journal 7, Rafsanjani, Marjan Kuchaki, and Amid Khatibi Bardsiri. A new heuristic approach for scheduling independent tasks on heterogeneous computing systems. International Journal of Machine Learning and Computing 2, no. 4, pp. 371-376.

[15] Rafsanjani, Marjan Kuchaki, et al. 2012, A new heuristic approach for scheduling independent tasks on heterogeneous computing systems. International Journal of Machine Learning and Computing 2, no. 4, pp. 371-376.

[16] Alharbi, Fahd, and K. Rabigh. 2012, Simple scheduling algorithm with load balancing for grid computing. Asian Transactions on Computers 2, no. 2.

[17] Hesam Izakian, Ajith Abraham, et al. 2009, Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In Computational Sciences and Optimization, CSO 2009, International Joint Conference, pp. 8-12, IEEE.

[18] Jinquan, Zhang, Ni Lina, et al. 2005, A heuristic scheduling strategy for independent tasks on grid. In High-Performance Computing in Asia-Pacific Region, 2005. Proceedings. Eighth International Conference on, pp. 6-pp. IEEE.

[19] Etminani, Kobra, et al. 2007, A Min-Min Max-Min selective algorithm for grid task scheduling. In Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on, pp. 1-7. IEEE.

[20] Naglaa M. Reda, A. Tawfik, et al. 2014, Sort-Mid tasks scheduling algorithm in grid computing. Journal of Advanced Research.,ELESEVIER.

[21] Chuliang Weng and Xinda Lu. 2005, Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid. Future Generation Computer Systems 21, All rights reserved, pp. 271–280 in Elsevier B.V.

[22] Jinquan, Zhang, Ni Lina, and Jiang Changjun. 2005, A heuristic scheduling strategy for independent tasks on grid. In High-Performance Computing in Asia-Pacific Region, 2005. Proceedings. Eighth International Conference on, pp. 6-pp. IEEE.

[23] Buyya, Rajkumar, and Manzur Murshed. 2002, Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing." Concurrency and computation: practice and experience 14, pp. 1175-1220.