A comparative study on seven static mapping heuristics for grid scheduling problem

Article *in* International Journal of Software Engineering and its Applications · January 2012

CITATIONS
15

READS
53

2 authors:

Amid Khatibi Bardsiri
Islamic Azad University
46 PUBLICATIONS **378** CITATIONS
SEE PROFILE

Seyyed Mohsen Hashemi
93 PUBLICATIONS **597** CITATIONS
SEE PROFILE

A Comparative Study on Seven Static Mapping Heuristics for Grid Scheduling Problem

Amid Khatibi Bardsiri[1] and Seyyed Mohsen Hashemi[2]

[1] *Bardsir Branch, Islamic Azad University, Kerman, IRAN*

[2]*Dean of the Software Engineering and Artificial Intelligence Department, Science and Research Branch, Islamic Azad University, Tehran, IRAN*

Abstract

Grid computing is a promising technology for future computing platforms and is expected to provide easier access to remote computational resources that are usually locally limited. Scheduling is one of the core steps to efficiently exploit the capabilities of grid computing (GC) systems. The problem of optimally mapping (defined as matching and scheduling) tasks onto the machines of a grid computing environment has been shown, in general, to be NP-complete, requiring the development of heuristic techniques. The efficient scheduling of independent tasks in a heterogeneous computing environment is an important problem in domains such as grid computing. Different criteria can be used for evaluating the efficiency of scheduling algorithms, the most important of which are makespan, resource utilization and matching proximity. In this paper we will compare 7 popular heuristics for statically mapping independent tasks onto grid computing systems.

Keywords: *Grid computing, Scheduling, Makespan, Resource utilization, Matching proximity*

1. Introduction

Grid computing and distributed computing, dealing with large scale and complex computing problems, is a hot topic in the computer science and research. According to Foster and Keselman in [1], grid computing is hardware and software infrastructure which offer a cheap, distributable, coordinated and reliable access to powerful computational capabilities. GC environments are well suited to meet the computational demands of large, diverse groups of tasks. A heterogeneous computing system consists of diversely capable machines harnessed together to execute a set of tasks that vary in their computational requirements [2]. To exploit the different capabilities of a suite of grid resources, typically a resource management system (RMS) allocates the resources to the tasks and the tasks are ordered for execution on the resources [3, 4]. Optimally scheduling is mapping a set of tasks to a set of resources to efficiently exploit the capabilities of such systems and is one of the key problems in grid computing environments. As mentioned in [5, 6] optimal mapping tasks to machines of a grid computing environment is an NP-complete problem and therefore the use of heuristics is one of the suitable approaches. Heuristics are needed to map (match and schedule) tasks onto machines in a GC system so as to optimize some figure of merit. The heuristics applied to the grid scheduling problem include Min-min, Max-min, Longest job to fastest resource-Shortest job to fastest resource (LJFR-SJFR), Sufferage, Work queue and others [3, 5, 7, 8, 9, 10]. Different criteria can be used for evaluating the efficiency of scheduling algorithms, the most important of which are makespan, resource utilization and matching proximity. Makespan is the time when grid computing system finishes the latest task.

Heuristics developed to perform this mapping function (Tasks scheduling) are often difficult to compare because of different underlying assumptions in the original study of each heuristic [11]. To facilitate these comparisons, certain simplifying assumptions were made. This paper presents a comparison of heuristic algorithms for static mapping to achieve better performance. The heuristic approaches were tested using the benchmark model of Braun et al [5].

2. Heuristic Descriptions

This section provides the description of 7 popular heuristics for mapping tasks to available machines in grid computing environments. Then we compare these heuristics.

2.1. OLB

Opportunistic Load Balancing (OLB) assigns each task, in arbitrary order, to the next machine that is expected to be available, regardless of the task's expected execution time on that machine. The intuition behind OLB is to keep all machines as busy as possible [5, 12].

2.2. MCT

Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task. This causes some tasks to be assigned to machines that do not have the minimum execution time for them [13].

2.3. Min-min

Min-min heuristic uses minimum completion time (MCT) as a metric, meaning that the task which can be completed the earliest is given priority. This heuristic begins with the set *U* of all unmapped tasks.

Then the set of minimum completion times (M), is found.

$$M = \begin{Bmatrix} min\left(completion\_time\left(T_i, M_j\right)\right) \mid T_i \in U \\ 1 \le i \le n \quad , \quad 1 \le j \le m \end{Bmatrix}$$

M consists of one entry for each unmapped task. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine and the workload of the selected machine will be updated. And finally the newly mapped task is removed from U and the process repeats until all tasks are mapped [5, 14].

2.4. Max-min

The Max-min heuristic is very similar to min-min and its metric is MCT too. It begins with the set *U* of all unmapped tasks. Then, the set of minimum completion times (M) is found as mentioned in previous section. Next, the task with the overall maximum completion time from M is selected and assigned to the corresponding machine and the workload of the selected machine will be updated. And finally the newly mapped task is removed from U and the process repeats until all tasks are mapped [5, 15].

2.5. LJFR-SJFR

Longest Job to Fastest Resource-Shortest Job to Fastest Resource (LJFR-SJFR) heuristic begins with the set U of all unmapped tasks. Then the set of minimum completion times is

found the same as Min-min. Next, the task with the overall minimum completion time from M is considered as the shortest job in the fastest resource (SJFR). Also the task with the overall maximum completion time from M is considered as the longest job in the fastest resource (LJFR). At the beginning, this method assigns the m longest tasks to the m available fastest resources (LJFR). Then this method assigns the shortest task to the fastest resource and the longest task to the fastest resource alternatively. After each allocation, the workload of each machine will be updated [3, 16].

2.6. Sufferage

In this heuristic for each task, the minimum and second minimum completion time are found in the first step. The difference between these two values is defined as the sufferage value. In the second step, the task with the maximum sufferage value is assigned to the corresponding machine with minimum completion time. Sufferage heuristic is based on the idea that better mappings can be generated by assigning a machine to a task that would "suffer" most in terms of expected completion time if that particular machine is not assigned to it [3, 7].

2.7. Maxstd

In Maxstd (Maximum Standard Deviation) heuristic, the task having the highest standard deviation of its expected execution time is scheduled first. The task having low standard deviation of task execution has less variation in execution time on different machines and hence, its delayed assignment for scheduling will not affect overall makespan much. Therefore the task having high standard deviation is assigned to the machine which has minimum completion time [10].

3. Scheduling Problem Definition

A grid computing environment is composed of computing resources where these resources can be a single PC, a cluster of workstations or a supercomputer. The main goal of the task scheduling in grid computing environments is the efficiently allocating tasks to machines. Tasks are originated from different users/applications, are independent. We use the ETC (Expected Time to Compute) matrix model introduced by Shoukat et al. for formulating the problem [17]. It is assumed that an accurate estimate of the expected execution time for each task on each machine is known prior to execution and contained within an ETC matrix. Each row of the ETC matrix contains the estimated execution times for a given task on each machine. Similarly, each column of the ETC matrix consists of the estimated execution times of a given machine for each task. $ETC[i,j]$ is the expected execution time of task i in machine j. For the simulation studies, characteristics of the ETC matrices were varied in an attempt to represent a range of possible GC environments. The amount of variance among the execution times of tasks in the meta-task for a given machine is defined as task heterogeneity. Machine heterogeneity represents the variation that is possible among the execution times for a given task across all the machines. To further vary the characteristics of the ETC matrices in an attempt to capture more aspects of realistic mapping situations, different ETC matrix consistencies were used. An ETC matrix is said to be consistent if whenever a machine m_j executes any task t_i faster than machine m_k, then machine m_j executes all tasks faster than machine m_k. Consistent matrices were generated by sorting each row of the ETC matrix independently. In contrast, inconsistent matrices characterize the situation where machine m_j may be faster than machine m_k for some tasks and slower for others. These matrices are left

in the unordered. Partially-consistent matrices are inconsistent matrices that include a consistent sub-matrix of a predefined size. For the partially-consistent matrices used here, the row elements in column positions [0, 2, 4, ...] of row i are extracted, sorted, and replaced in order, while the row elements in column positions [1, 3, 5, ...] remain unordered (i.e., the even columns are consistent and the odd columns are, in general, inconsistent). Therefore twelve combinations of ETC matrix characteristics were used: high or low task heterogeneity, high or low machine heterogeneity, and one type of consistency (consistent, inconsistent, or partially consistent).

4. Performance Evaluation

This section, after describing the benchmark problems, provides the comparison of 7 heuristics for mapping tasks to available machines in grid computing environments. For each heuristic and each type of ETC matrix, the results were averaged over 100 different ETC matrices of the same type (i.e., 100 mappings). The experimental results discussed below were obtained on a 2 GHz processor with 2 GB of RAM. These heuristics are implemented using Matlab Environment and run on 12 different types of ETC matrices.

4.1. Simulation Model

In this paper, we used the benchmark proposed in [5]. The simulation model is based on expected time to compute (ETC) matrix for 512 tasks and 16 machines. The instances of the benchmark are classified into 12 different types of ETC matrices according to the three following metrics: task heterogeneity, machine heterogeneity, and consistency. Instances consist of 512 tasks and 16 machines and are labeled as x-yy-zz as follows:

- x shows the type of inconsistency; c means consistent, i means inconsistent, and p means partially-consistent.

- yy indicates the heterogeneity of the tasks; hi means high and lo means low.

- zz represents the heterogeneity of the machines; hi means high and lo means low.

For example, c-lohi means low heterogeneity in tasks, high heterogeneity in machines, and consistent environment.

4.2. Makespan

A meta-task is defined as a collection of independent task (i.e. task doesn't require any communication with other tasks) [11]. Tasks derive mapping statically. For static mapping, the number of tasks, n and the number of machines, m is known a priori. Assume that $C_{i,j}$ ($i \in \{1,2,...,n\}$, $j \in \{1,2,..., m\}$) is the completion time for performing *i*th task in *j*th machine and W_j ($j \in \{1,2,...,m\}$) is the previous workload of M_j, then Eq. (1) shows the time required for M_j to complete the tasks included in it. According to the aforementioned definition, makespan can be estimated using Eq. (2) [18, 19, 20].

$$\sum C_j + W_j \qquad (1)$$

$$Makespan = \max_{j \in 1,...,m} \{\sum C_j + W_j\} \qquad (2)$$

Among most popular and extensively studied optimization criterion is the minimization of the makespan. Small values of makespan mean that the scheduler is providing good and efficient planning of tasks to resources. The obtained makespan using mentioned heuristics

are compared in Table 1. The results from the simulations for selected cases of consistency, task heterogeneity, and machine heterogeneity are shown in Figures 1 through 4. The results are obtained as an average of 100 simulations.

4.3. Resource Utilization

Maximizing the resource utilization of the GC system is another important objective. This criterion is gaining importance due to the economic aspects of GC systems. The heuristics should improve the utilization of resources by reducing the idle time of the machines. One possible definition of this parameter is to consider the average utilization of resources. For instance, in the ETC model, it can be defined as follows [21]:

$$Utilization = \frac{\sum_{\{i \in Machines\}} C_i}{makespan \times nb\_machines}$$

And we aim at maximizing this value over all possible schedules. Table 2 shows the resource utilization values for mentioned heuristics and Figure 5 shows the graphical representation of Table 2.

Table 1. Comparison of makespan values obtained by Heuristics

Instance	OLB	MCT	Min-min	Max-min	Sufferage	LJFR-SJFR	Maxstd
c_hihi	1.48E+07	1.13E+07	8.38E+06	1.20E+07	1.00E+07	1.20E+07	1.09E+07
c_hilo	1.96E+05	1.58E+05	1.33E+05	1.80E+05	1.37E+05	1.75E+05	1.53E+05
c_lohi	4.99E+05	3.83E+05	2.81E+05	4.05E+05	3.38E+05	4.04E+05	3.67E+05
c_lolo	6.58E+03	5.30E+03	4.50E+03	6.05E+03	4.65E+03	5.87E+03	5.14E+03
i_hihi	2.51E+07	4.28E+06	3.58E+06	7.25E+06	3.30E+06	6.34E+06	3.95E+06
i_hilo	2.76E+05	7.39E+04	6.65E+04	1.23E+05	6.24E+04	1.06E+05	6.96E+04
i_lohi	8.50E+05	1.45E+05	1.21E+05	2.46E+05	1.10E+05	2.15E+05	1.33E+05
i_lolo	9.38E+03	2.47E+03	2.21E+03	4.11E+03	2.10E+03	3.55E+03	2.35E+03
p_hihi	2.04E+07	6.23E+06	4.86E+06	9.27E+06	5.08E+06	8.38E+06	5.80E+06
p_hilo	2.38E+05	9.82E+04	8.43E+04	1.48E+05	8.24E+04	1.31E+05	9.36E+04
p_lohi	6.82E+05	2.10E+05	1.64E+05	3.12E+05	1.70E+05	2.82E+05	1.95E+05
p_lolo	7.97E+03	3.32E+03	2.83E+03	4.96E+03	2.76E+03	4.38E+03	3.16E+03

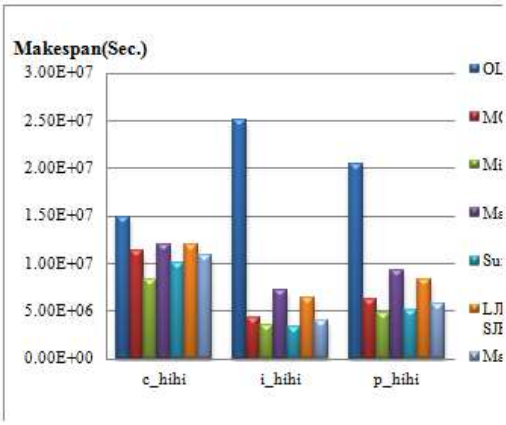

Figure 1. Makespan value, high task, high machine heterogeneity

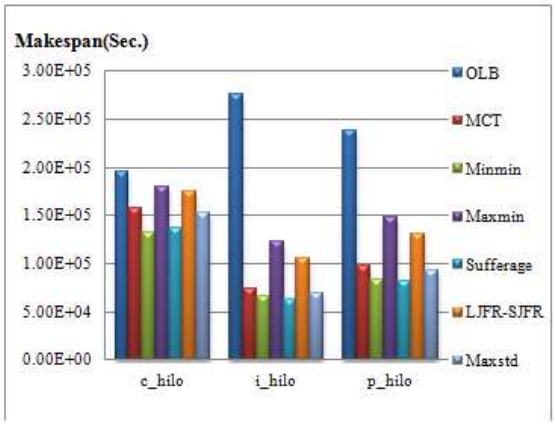

Figure 2. Makespan value, high task, low machine heterogeneity

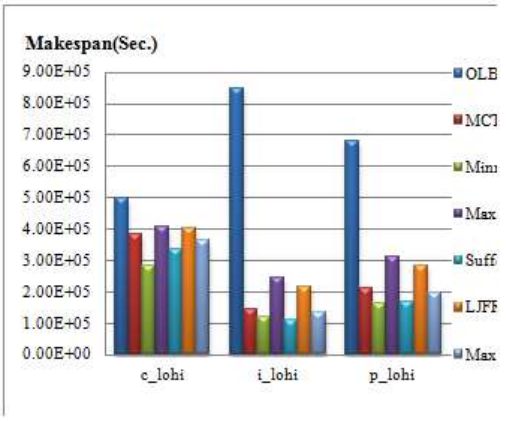

Figure 3. Makespan value, low task, high machine heterogeneity

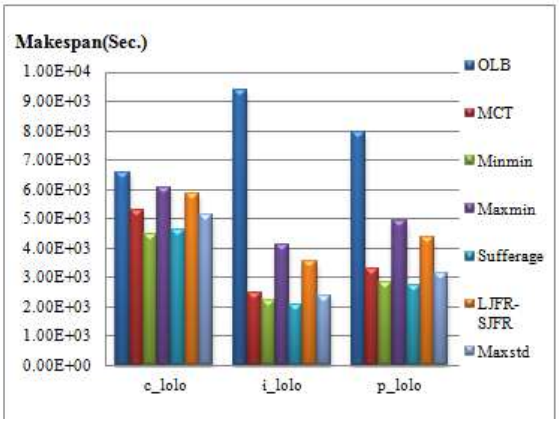

Figure 4. Makespan value, low task, low machine heterogeneity

4.4. Matching proximity

We consider also a third parameter, called matching proximity, which is very useful for measuring the performance of the presented methods. Matching proximity indicates the degree of proximity of a given schedule with regard to the schedule produced by Minimum Execution Time (MET) method. In contrast to OLB, MET assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability. The motivation behind MET is to give each task to its best machine. Therefore, a large value of matching proximity means that a large number of jobs is assigned to the machine that executes them faster. Formally, this parameter is defined as follows [22]:

$$matching\_proximity = \frac{\sum ETC[i][S[i]]}{\sum ETC[i][MET[i]]}$$

Table 3 represents the matching proximity values obtained by heuristics. Graphical representation of Table 3 shows in Figure 6.

Table 2. Comparison Results on Resource Utilization

Instance	OLB	MCT	Min-min	Max-min	Sufferage	LJFR-SJFR	Maxstd
c_hihi	0.9231	0.9479	0.8908	0.9993	0.9553	0.9693	0.9916
c_hilo	0.9367	0.9581	0.9235	0.9995	0.9672	0.9772	0.9954
c_lohi	0.9225	0.9479	0.8874	0.9992	0.9697	0.9697	0.9975
c_lolo	0.9401	0.9583	0.9253	0.9994	0.9885	0.9782	0.9965
i_hihi	0.9532	0.9226	0.8257	0.9982	0.9371	0.9841	0.9869
i_hilo	0.9572	0.9481	0.8943	0.9988	0.9474	0.9834	0.9946
i_lohi	0.9525	0.918	0.8368	0.9981	0.9394	0.9845	0.9844
i_lolo	0.9551	0.9512	0.8955	0.9988	0.9492	0.9836	0.9931
p_hihi	0.9439	0.9333	0.8472	0.9987	0.9571	0.9814	0.9940
p_hilo	0.9481	0.9522	0.9009	0.9991	0.9741	0.9816	0.9961
p_lohi	0.9407	0.9369	0.8469	0.9987	0.9596	0.9811	0.9917
p_lolo	0.9489	0.9508	0.9048	0.9990	0.9733	0.9821	0.9966

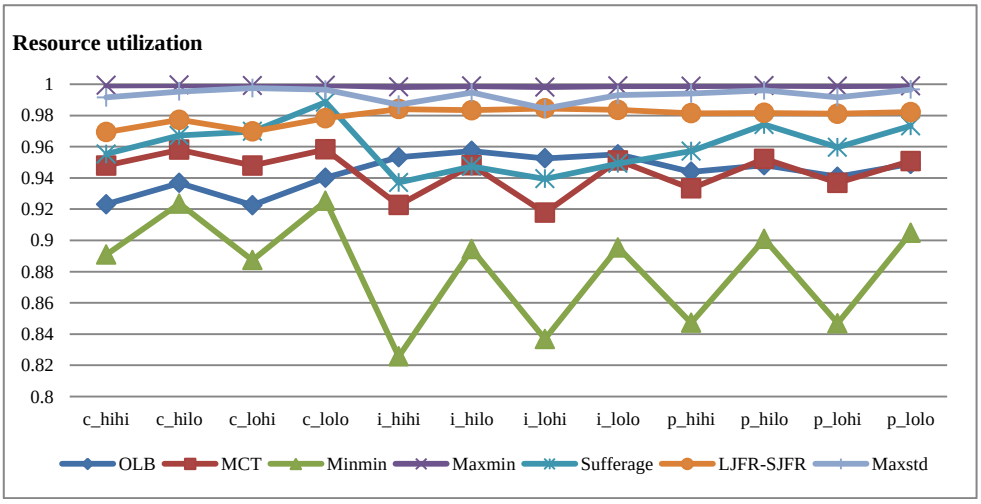

Figure 5. Graphical representation of Table 2

Table 3. Comparison of Matching Proximity Values obtained by Heuristics

Instance	OLB	MCT	Min-min	Max-min	Sufferage	LJFR-SJFR	Maxstd
c_hihi	0.209	0.266	0.385	0.228	0.291	0.245	0.261
c_hilo	0.319	0.389	0.479	0.327	0.432	0.341	0.374
c_lohi	0.206	0.261	0.38	0.234	0.287	0.242	0.261
c_lolo	0.316	0.386	0.471	0.324	0.427	0.342	0.353
i_hihi	0.119	0.717	0.957	0.391	0.919	0.454	0.713
i_hilo	0.222	0.844	0.986	0.478	0.981	0.558	0.825
i_lohi	0.118	0.718	0.962	0.389	0.927	0.448	0.732
i_lolo	0.220	0.837	0.993	0.479	0.996	0.561	0.852
p_hihi	0.148	0.488	0.692	0.307	0.584	0.344	0.504
p_hilo	0.259	0.628	0.773	0.394	0.731	0.454	0.607
p_lohi	0.149	0.493	0.703	0.309	0.592	0.347	0.479
p_lolo	0.261	0.626	0.772	0.396	0.732	0.457	0.597

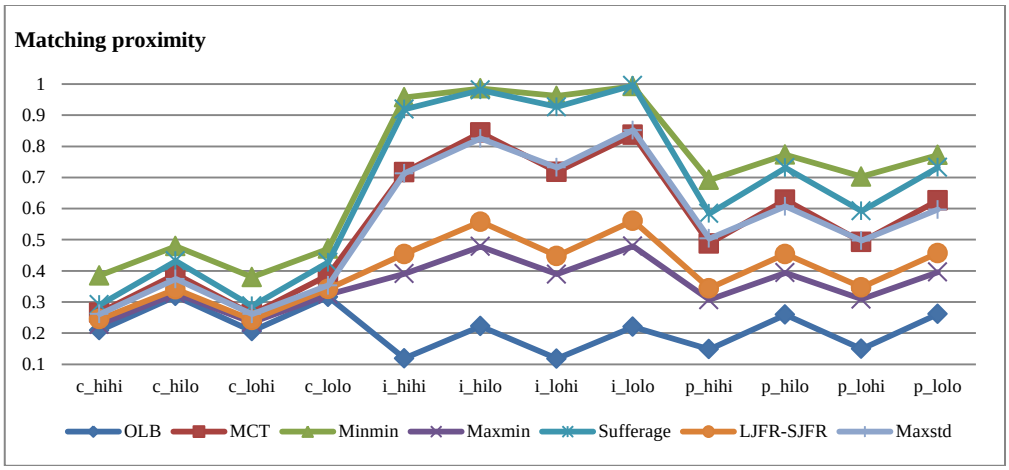

Figure 6. Graphical Representation of Table 3

5. Conclusions and Future Work

Task Scheduling is a critical design issue of distributed computing. A computational grid is a highly distributed environment. Selecting the appropriate machine for the specific task is one of the challenging works in computational grids. In this paper we compare 7 heuristics for scheduling in GC environments. The goal of the scheduler in this paper is minimizing makespan and maximizing resources utilization and matching proximity. The goal of this study was to provide a basis for comparison and insights into circumstances when one static technique will out-perform another for 7 different heuristics. In our future work, we plan to implement a scheduler for dependent task scheduling and to find other methods for neighborhood search in order to improve the efficiency of scheduler.

References

[1] I. Foster and C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, San Francisco, **(2003)**.
[2] S. Ali, T. D. Braun, H. J. Siegel and A. A. Maciejewski, "Heterogeneous Computing", Encyclopedia of Distributed Computing, Kluwer Academic, **(2001)**.
[3] H. Izakian, A. Abraham and V. Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments", Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO), vol. 1, **(2009)**, pp. 8-12.
[4] M. Tracy, T. D. Braun and H. Siegel, "High-performance Mixed-machine Heterogeneous Computing", 6th Euro-micro Workshop on Parallel and Distributed Processing, **(1998)**, pp. 3-9.
[5] R. Braun, H. J. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing (JPDC), vol. 61, no. 6, **(2001)**, pp. 810-837.
[6] D. Fernandez-Baca, "Allocating Modules to Processors in a Distributed System", IEEE Transaction, Software Engineering, vol. 15, no. 11, **(1989)**, pp. 1427–1436.
[7] E. U. Munir, L. Jian-Zhong, S. Sheng-Fei and Q. Rasool, "Performance Analysis of Task Scheduling Heuristics in Grid", Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC), vol. 6, **(2007)**, pp. 3093-3098.
[8] H. Baghban and A. M. Rahmani, "A Heuristic on Job Scheduling in Grid Computing Environment", Proceedings of the 7th IEEE International Conference on Grid and Cooperative Computing (GCC), **(2008)**, pp. 141-146.

[9] M. Macheswaran, 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 Distributed Computing (JPDC), vol. 59, no. 2, **(1999)**, pp. 107-131.
[10] E. U. Munir, J. Li, S. Shi, Z. Zou and D. Yang, "MaxStd: A Task Scheduling Heuristic for Heterogeneous Computing Environment", Information Technology Journal, vol. 7, no. 4, **(2008)**, pp. 679-683.
[11] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys and B. Yao, "A Taxonomy for Describing Datching and Scheduling Heuristics for Mixed-machine Heterogeneous Computing Systems", Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, **(1998)**, pp. 330-335.
[12] R. Armstrong, D. Hensgen and T. Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions", Proceedings of the 7th IEEE Heterogeneous Computing Workshop (HCW '98), **(1998)**, pp. 79-87.
[13] F. Xhafa, L. Barolli and A. Durresi, "Immediate Mode Scheduling in Grid Systems", International Journal of Web and Grid Services (IJWGS), vol. 3, no. 2, **(2007)**, pp. 219-236.
[14] R. F. Freund and M. Gherrity, "Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net", Proceedings of the 7th IEEE Heterogeneous Computing Workshop (HCW '98), **(1998)**, pp.184-199.
[15] R. F. Freund and H. J. Siegel, "Heterogeneous Processing", IEEE Computer, vol. 26, no. 6, **(1993)**, pp. 13-17.
[16] A. Abraham, R. Buyya and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids", Proceedings of the International Conference on Advanced Computing and Communications, **(2000)**.
[17] S. Ali, H. J. Siegel, M. Maheswaran and D. Hensgen, "Modeling Task Execution Time Behavior in Heterogeneous Computing Systems", 9th IEEE Heterogeneous Computing Workshop (HCW 2000), **(2000)**, pp. 185-199.
[18] H. Yan, X. Q. Shen, X. Li and M. Huiwu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing", Proceedings of the IEEE 4th International Conference on Machine Learning and Cybernetics (ICMLC), vol. 5, **(2005)**, pp. 2957-2961.
[19] J. Carretero and F. Xhafa, "Using Genetic Algorithms for Scheduling Jobs in Large Scale Grid Applications", Journal of Technological and Economic Development of Economy (Technol Econ Dev Econ J.), A Research Journal of Vilnius Gediminas Technical University, vol. 12, no. 1, **(2006)**, pp. 11-17.
[20] F. Xhafa, L. Barolli and A. Durresi, "Batch Mode Scheduling in Grid Systems", International Journal of Web and Grid Services (IJWGS), vol. 3, no. 1, **(2007)**, pp. 19-37.
[21] F. Xhafa and A. Abraham, "Computational Models and Heuristic Methods for Grid Scheduling Problems", Future Generation Computer Systems (FGCS), vol. 26, **(2010)**, pp. 608-621.
[22] F. Xhafa and A. Abraham, "Meta-heuristics for Grid Scheduling Problems", In Meta-heuristics for Scheduling in Distributed Computing Environments, vol. 146, Springer, **(2008)**, pp. 1-37.