A New Heuristic Approach for Scheduling Independent Tasks on Heterogeneous Computing Systems

Marjan Kuchaki Rafsanjani and Amid Khatibi Bardsiri

***Abstract*—Scheduling is one of the core steps to efficiently exploit the capabilities of heterogeneous computing systems. The problem of mapping meta-tasks to a machine is shown to be NP-complete. The NP-complete problem can be solved only using heuristic approach. There are a number of heuristic algorithms that were tailored to deal with scheduling of independent tasks. Different criteria can be used for evaluating the efficiency of scheduling algorithms. The most important of them are makespan, flowtime and resource utilization. In this paper, a new heuristic algorithm for scheduling meta-tasks in heterogeneous computing system is presented. The proposed algorithm improves the performance in both makespan and effective utilization of resources by reducing the idle time of the machine. The performance analyses show that the proposed algorithm has a better resource utilization rate and reduced makespan than the other known algorithms.**

***Index Terms*—ETC matrix, Flowtime, Heterogeneous Computing (HC), Makespan.**

I. Introduction

Grid computing and distributed computing, dealing with large scale and complex computing problems, is a hot topic in the computer science and research. Mixed-machine heterogeneous computing (HC) environments utilize a distributed suite of different machines, interconnected with computer network, to perform different computationally intensive applications that have diverse requirements [1]. Miscellaneous resources should be orchestrated to perform a number of tasks in parallel or to solve complex tasks atomized to variety of independent subtasks [2]. To exploit the different capabilities of a suite of heterogeneous resources, a resource management system (RMS) allocates the resources to the tasks and the tasks are ordered for execution on the resources. At a time interval in HC environment a number of tasks are received by RMS. Task scheduling is mapping a set of tasks to a set of resources to efficiently exploit the capabilities of such. It has been shown, that an optimal mapping of computational tasks to available machines in an HC suite is a NP-complete problem [3] and as such, it is a subject to various heuristic and meta-heuristic algorithms.

The heuristics applied to the task scheduling problem include Min-min heuristic, Max-min heuristic, longest job to fastest resource- shortest job to fastest resource (LJFR-SJFR) heuristic, Sufferage heuristic, Work queue heuristic and others [1, 4, 5]. Different criteria can be used for evaluating the efficiency of scheduling algorithms. The most important of them are makespan, flowtime and resource utilization. Makespan is the time when HC system finishes the latest task, and flowtime is the sum of execution times of all the tasks. An optimal scheduling will be the one that minimizes the flowtime and makespan. The objectives of scheduling algorithm are increasing the system throughput measure [6, 7], reducing task completion time, better resource utilization rate and balancing the load well. This paper presents a new scheduling algorithm for static mapping to achieve better performance. The proposed heuristic approach was tested using the benchmark model of Braun et al [1].

Manuscript received May 27; revised June 26, 2012.

Marjan Kuchaki Rafsanjani (Corresponding author) is with the Department of Computer Science, Shahid Bahonar University of Kerman. Kerman, Iran, Postal code: 76169-14111 (e-mail: kuchaki@uk.ac.ir).

Amid Khatibi Bardsiri is with the Bardsir branch, Islamic Azad University, Kerman, Iran (e-mail: a.khatibi@srbiau.ac.ir).

II. Related Work

A set of heuristic algorithms has been designed to schedule meta-tasks to heterogeneous computing systems. It is assumed that the heuristic derive mapping statically for the collection of independent meta-task. The scheduling problem is computationally hard even though there are no data dependencies among the jobs.

A. 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 [8, 9].

B. MET

In contrast to OLB, Minimum Execution Time (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. This can cause a severe load imbalance across machines.

C. 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 [1].

D. 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.

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

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 [1, 9, 10].

E. 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 [1, 9].

F. LJFR-SJFR

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 [4, 11].

G. 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 [4, 12].

H. Work Queue

This heuristic is a straightforward and adaptive scheduling algorithm for scheduling sets of independent tasks. In this method, the heuristic selects a task randomly and assigns it to the machine as soon as it becomes available (in other word, machine with minimum workload) [4, 13].

I. Maximum Standard Deviation

In Maxstd 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. Hence the task having high standard deviation is assigned to the machine which has minimum completion time [14].

J. SA

Simulated Annealing (SA) is an iterative technique that considers only one possible mapping for each meta-task at a time. Simulated annealing uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space based on a system temperature [15].

K. GSA

The Genetic Simulated Annealing (GSA) heuristics is a combination of the GA and SA heuristics. GSA follows the procedures similar to the GA. For the selection process, GSA uses the SA cooling schedule and system temperature [16].

L. Tabu

Tabu search is a solution space search that keeps track of the regions of the solution space to avoid repeating a search near the areas that have already been searched. A mapping of meta-tasks uses the same representation as a chromosome in the GA approach. The implementation of Tabu search begins with a random mapping, generated from a uniform distribution [17].

Though the above stated heuristic algorithms have advantages, they do have their own disadvantages. OLB leads to poor makespan since it does not consider the expected execution time while mapping the meta-tasks to the machines and it is also hard to achieve dynamic load balance of jobs. MET results in severe load imbalance across the machines. The experimental results from [1, 4, 12, 18, 19] show that OLB, MET, Work Queue, SA, GSA and Tabu do not produce good mappings. In contrast, others are able to deliver good performance and more, only they are intended. Fig. 1 shows the geometric mean of makespan for the 10 considered cases using Braun et al. benchmark described in section IV. Among the stated algorithms, Min-min is the simple and fastest algorithm.

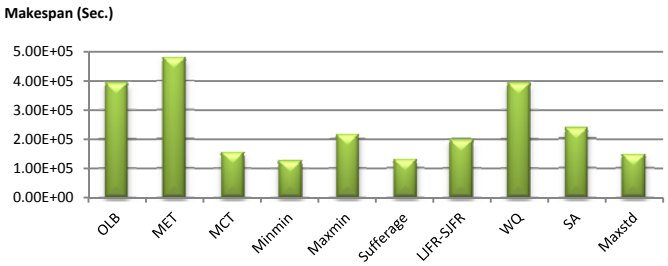

Fig. 1. Makespan obtained by heuristic algorithms

III. Problm Definition

A heterogeneous 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 HC environments is the efficiently allocating tasks to machines. The efficiency means that we are interested to allocate tasks as fast as possible and optimizing the makespan and execution time objectives. 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 [20]. 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 heterogeneous environments. Using the ETC matrix model, the scheduling problem can be defined as follows:

- A number of independent tasks to be allocated to the available resources. Because of Non-preemptive scheduling, each task has to be processed completely in a single machine.
- Number of machines is available to participate in the allocation of tasks.
- The workload of each task (in millions of instructions)
- The Computing capacity of each resources (in MIPS)
- Ready[m] represents the ready time of the machine after completing the previously assigned tasks.
- ETC matrix of size $n \times m$, where n represents the number of tasks and m represents the number of machines.

A. The Proposed Algorithm

A meta-task is defined as a collection of independent task (i.e. task doesn't require any communication with other tasks) [21]. 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. (2) shows the time required for M_j to complete the tasks included in it. According to the aforementioned definition, makespan and flowtime can be estimated using Eq. (3) and Eq. (4) respectively [4].

$$\sum C_i + W_i \tag{2}$$

$$Makespan = \max_{j \in 1,...,m} \{\sum C_j + W_j\} \tag{3}$$

$$Flowtime = \sum_{i=1}^{m} E_{ij} \qquad i \in 1...n \tag{4}$$

The steps of the proposed algorithm are as follows: In the first step, heuristic begins with the set of all unmapped tasks. Then the set of minimum completion times is found, same as Min-min heuristic. In the second step, for each task, the minimum, second minimum completion time and minimum execution time are found. Then the difference between these two minimum completion time values is multiplied by the amount of minimum completion time and minimum execution time divided. The task which its value is maximum, will be selected and remove from set of unmapped tasks. The sequence of the execution of the proposed heuristic scheduling algorithm is as follows:

(1) While there are tasks to schedule
(2) U=Empty
(3) For all $Task_i$ to schedule
$Machine_i$
(4) For all
(5) $c_{ij} = W_i + E_{ij}$
(6) End for
(7) $C_{ix} = \min(C_{ij}) \qquad \forall j = 1,..., m$
(8) $C_{iy} = Sec.\min(C_{ij})$
(9) $E_{iz} = \min(E_{ij}) \qquad \forall j = 1,..., m$
(10) $k_{ix} = (C_{iy} - C_{ix}) \times (\frac{E_{iz}}{C_{ix}})$
(11) $U = U \bigcup k_{ix}$
(12) End for
(13) Select the maximum k value from U set
(14) Allocate related task and update workload
(15) Delete task from unmapped tasks
(16) End while

The intuition behind this heuristic is that select pair machines and tasks from the first step that the machine can executes its corresponding task effectively with a lower execution time in comparison with other machines and also, the assistance of the suffer value makes the algorithm stronger. The results discussed below were obtained on a PC with 2 GHz processor and 2 GB of RAM. Proposed heuristic has asymptotic complexity of $O(mn^2)$, as the Min-min, and executed for less than 1s per ETC matrix.

IV. Experimental Results

In this section, after the benchmark description, various heuristic scheduling algorithms were compared with the proposed algorithm. These heuristics are implemented using Matlab environment and run on 12 different types of ETC matrices. 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).

A. Benchmark Description

In this paper, we used the benchmark proposed in [1]. 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. In ETC matrix, the amount of variance among the execution times of tasks 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. Also an ETC matrix is said to be consistent whenever a machine M_j executes any task T_i faster than machine M_k; in this case, machine M_j executes all tasks faster than machine M_k. 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. Partially-consistent matrices are inconsistent matrices that include a consistent sub-matrix of a predefined size. Instances consist of 512 tasks and 16 machines and are labeled as u-x-yy-zz as follows:

- u means uniform distribution used in generating the matrices
- 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, u-c-lohi means low heterogeneity in tasks, high heterogeneity in machines, and consistent environment.

B. Makespan and Flowtime

The obtained makespan and flowtime using mentioned heuristics are compared in Tables II and III respectively; in each instance the best cases are bold. The results are obtained as an average of 100 simulations. Fig. 2 and Fig. 3 show the geometric mean of makespan and flowtime for the 12 considered cases. Finally, Fig. 4 shows the graphical representation of Table II. 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. Another important optimization criterion is that of flowtime, which refers to the response time to the user submissions of task executions. Minimizing the value of flowtime means reducing the average response time of the HC system. In general, the makespan value is more important. As shown in Fig. 2 and Fig. 3, the new algorithm is better than all in makespan value and second best in flowtime value. Min-min gave the second best result (after new algorithm) in makespan and best in flowtime value. However the drawback of Min-min is that, it is unable to balance the load because it usually assigns the small task first and few larger tasks, while at the same time, several machines sit idle, which leads to poor utilization of resources. The proposed algorithm retains the advantage of Min-min and reduces the idle time of the resources, which in turn leads to better makespan. In the next section, this case is explained completely and the important advantage of the new algorithm is seen.

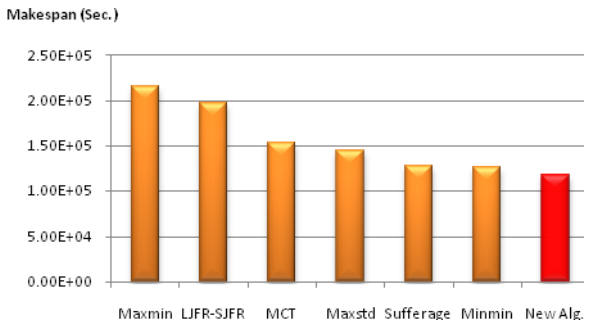

Fig. 2. Comparison results between heuristics on makespan

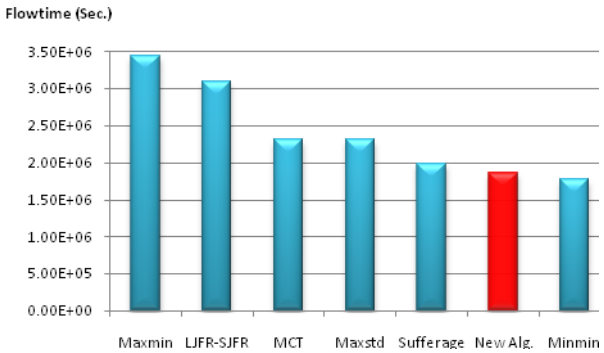

Fig. 3. Comparison results between heuristics on flowtime

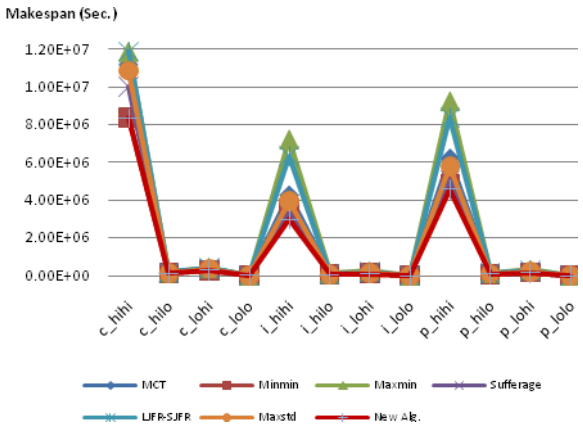

Fig. 4. Graphical representation of Table II

C. Resource Utilization

Maximizing the resource utilization of the HC system is another important objective. This criterion is gaining importance due to the economic aspects of HC 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:

$$Utilization = \frac{\sum_{\{i \in Machines\}} C_i}{makespan \times nb\_machines} \quad (5)$$

We aim at maximizing this value over all possible schedules [22]. Table I shows the mean of utilization values for mentioned heuristics. Be noted that the new algorithm among all has the best resource utilization rate.

TABLE I: COMPARISON RESULTS BETWEEN HEURISTICS ON RESOURCE UTILIZATION

Heuristic	Utilization
Maxstd	99.41%
MCT	94.52%
Min-min	88.15%
Max-min	99.59%
Sufferage	95.94%
LJFR-SJFR	98.08%
New Alg.	**99.68%**

Fig. 5 represents the improvement of new heuristic scheduling algorithm over Min-min in whole 12 different types of instances.

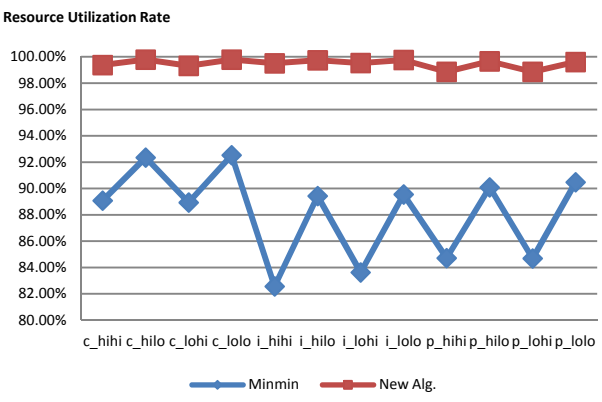

Fig. 5. Improvement of New Alg. over Min-min heuristic

TABLE II: COMPARISON OF MAKESPAN VALUES OBTAINED BY HEURISTICS

Instance	MCT	Min-min	Max-min	Sufferage	LJFR-SJFR	Maxstd	New Alg.
u_c_hihi	1.12E+07	8.37E+06	1.19E+07	1.00E+07	1.19E+07	1.09E+07	**8.36E+06**
u_c_hilo	1.57E+05	1.33E+05	1.79E+05	1.37E+05	1.75E+05	1.53E+05	**1.32E+05**
u_c_lohi	3.82E+05	2.80E+05	4.05E+05	3.38E+05	4.04E+05	3.67E+05	**2.79E+05**
u_c_lolo	5.29E+03	4.49E+03	6.05E+03	4.65E+03	5.87E+03	5.14E+03	**4.44E+03**
u_i_hihi	4.28E+06	3.58E+06	7.25E+06	3.30E+06	6.34E+06	3.95E+06	**3.00E+06**
u_i_hilo	7.39E+04	6.65E+04	1.23E+05	6.24E+04	1.06E+05	6.96E+04	**5.92E+04**
u_i_lohi	1.45E+05	1.21E+05	2.46E+05	1.10E+05	2.15E+05	1.33E+05	**1.01E+05**
u_i_lolo	2.47E+03	2.21E+03	4.11E+03	2.10E+03	3.55E+03	2.35E+03	**1.98E+03**
u_p_hihi	6.23E+06	4.86E+06	9.27E+06	5.08E+06	8.38E+06	5.80E+06	**4.58E+06**
u_p_hilo	9.82E+04	8.43E+04	1.48E+05	8.24E+04	1.31E+05	9.36E+04	**7.83E+04**
u_p_lohi	2.10E+05	1.64E+05	3.12E+05	1.70E+05	2.82E+05	1.95E+05	**1.54E+05**
u_p_lolo	3.32E+03	2.83E+03	4.96E+03	2.76E+03	4.38E+03	3.16E+03	**2.64E+03**

TABLE III: COMPARISON OF FLOWTIME VALUES OBTAINDE BY HEURISTICS

Instance	MCT	Min-min	Max-min	Sufferage	LJFR-SJFR	Maxstd	New Alg.
u_c_hihi	1.7094e+008	**1.1883e+008**	1.9197e+008	1.5689e+008	1.8573e+008	1.7335e+008	1.3615e+008
u_c_hilo	2.4158e+006	**1.9686e+006**	2.8764e+006	2.1709e+006	2.7575e+006	2.4401e+006	2.1064e+006
u_c_lohi	5.8140e+006	**3.9979e+006**	6.4841e+006	5.2858e+006	6.2666e+006	5.8540e+006	4.6213e+006
u_c_lolo	8.1220e+004	**6.6647e+004**	9.6831e+004	7.3496e+004	9.1862e+004	8.2116e+004	7.0869e+004
u_i_hihi	6.3365e+007	4.7425e+007	1.1603e+008	4.9445e+007	1.0079e+008	6.2433e+007	**4.7139e+007**
u_i_hilo	1.1190e+006	9.5098e+005	1.9686e+006	9.4654e+005	1.6823e+006	1.1082e+006	**9.4449e+005**
u_i_lohi	2.1333e+006	1.5982e+006	3.9375e+006	1.6523e+006	3.4131e+006	2.0919e+006	**1.5803e+006**
u_i_lolo	3.7673e+004	3.1721e+004	6.5717e+004	3.1699e+004	5.6180e+004	3.7414e+004	**3.1553e+004**
u_p_hihi	9.3119e+007	**6.5784e+007**	1.4850e+008	7.7887e+007	1.3203e+008	9.2170e+007	7.2519e+007
u_p_hilo	1.4961e+006	1.2129e+006	2.3715e+006	1.2884e+006	2.0642e+006	1.4925e+006	**1.2087e+006**
u_p_lohi	3.1362e+006	**2.1975e+006**	4.9998e+006	2.6098e+006	4.4470e+006	3.1015e+006	2.4356e+006
u_p_lolo	5.0352e+004	**4.0844e+004**	7.9454e+004	4.3095e+004	6.8994e+004	5.0333e+004	4.2096e+004

V. CONCLUSION

Scheduling in HC environments is an NP-complete problem. Therefore, using heuristic algorithms is a suitable approach in order to cope with its difficulty in practice. In this paper, we have presented a new scheduling algorithm for scheduling in HC environments. The goal of the scheduler in this paper is minimizing makespan, flowtime and maximizes resources utilization. The implementation of proposed heuristic scheduling algorithm and various existing algorithm are tested using the benchmark simulation model for distributed heterogeneous computing systems by Braun et al. The experimental results show that proposed algorithm performs better than the existing heuristic algorithm in various systems and settings and also it delivers improved makespan and resource utilization on various heterogeneous environments such as job heterogeneity (high, low), machine heterogeneity (high, low), and consistency (consistent, inconsistent, semi-consistent or partially-consistent). The future research will be directed towards the factors such as CPU workload, communication delay and so on.

REFERENCES

[1] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, and R. 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, vol. 61, no. 6, 2001, pp. 810-837.

[2] M. Tracy, T. D. Braun, and H. Siegel, "High-performance Mixed-machine Heterogeneous Computing," in *Proc. of the 6th Euro-micro Workshop on Parallel and Distributed Processing*, 1998, pp. 3-9.

[3] D. Fernandez-Baca, "Allocating Modules to Processors in a Distributed System," IEEE Trans., Software Engineering, 1989, pp. 1427–1436.

[4] H. Izakian, A. Abraham and V. Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments," in *Proc. of the International Joint Conference on Computational Sciences and Optimization*, IEEE, vol. 1, 2009, pp. 8-12.

[5] E .U. Munir, L. Jian-Zhong, Sh. Sheng-Fei, and Q. Rasool, "Performance Analysis of Task Scheduling Heuristics in Grid," in *Proc. of the International Conference on Machine Learning and Cybernetics(ICMLC)*, vol. 6, 2007, pp. 3093-3098.

[6] H. Baghban and A. M. Rahmani, "A Heuristic on Job Scheduling in Grid Computing Environment," in *Proc. of the seventh IEEE International Conference on Grid and Cooperative Computing*, 2008, pp. 141-146.

[7] Q. Zhang and L. Zhen, "Design of Grid Resource Management System Based on Divided Min-min scheduling Algorithm," in *Proc. of the IEEE First International Workshop on Education Technology and Computer Science*, 2009, pp. 613-618.

[8] R. Armstrong, D. Hensgen, and T. Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions," in *Proc. of the 7th IEEE Heterogeneous Computing Workshop (HCW '98)*, 1998, pp. 79-87.

[9] R. F. Freund and H. Siegel, "Heterogeneous Processing," IEEE Computer, vol. 26, no. 6, 1993, pp. 13-17.

[10] R. F. Freund and M. Gherrity, "Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net", in *Proc. of the 7th IEEE HCW*, 1998.

[11] A. Abraham, R. Buyya and B. Nath, "Nature's Heuristics for Scheduling Jobs on Computational Grids," in *Proc. of the* International *Conference on Advanced Computing and Communications*, 2000.

[12] 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, vol. 59, no. 2, 1999, pp. 107-131.

[13] T. Hagerup, "Allocating Independent Tasks to Parallel Processors," An Experimental Study, Journal of Parallel and Distributed Computing, vol. 47, 1997, pp. 185-197.

[14] E. U. Munir, "MaxStd: A Task Scheduling Heuristic for Heterogeneous Computing Enviroment," Information Technology Journal, 2008, ISSN-1812-5638.

[15] M. Coli and P. Palazzari, "Real Time Pipelined System Design through Simulated Annealing," Journal of Systems Architecture, vol. 42 no. 6-7, 1996, pp. 465-475.

[16] H. Chen, N.S. Flann, and D.W. Watson, "Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Approach," IEEE trans. on Parallel and Distributed Computing, vol. 9, no. 2, 1998, pp. 126-136.
[17] I. D. Falco, R. D. Balio, E. Tarantino, and R. Vaccaro, "Improving Search by Incorporating Evolution Principles in Parallel Tabu Search," in *Proc. of the IEEE Conference on Evolutionary Computation*, 1994, pp. 823-828.
[18] H. Yan, X-Q. Shen, X. Li, and M. Huiwu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing," in *Proc. of the IEEE 4th International Conference on Machine Learning and Cybernetics*, 2005, pp. 2957-2961.
[19] F. Xhafa, L. Barolli, and A. Durresi, "Immediate Mode Scheduling in Grid Systems," International Journal of Web and Grid Services, vol. 3, no. 2, 2007, pp. 219-236.
[20] S. Ali, H. J. Siegel, M. Maheswaran, and D. Hensgen, "Modeling Task Execution Time Behavior in Heterogeneous Computing Systems," Tamkang Journal Science and Engineering, 2000, pp. 195-207.
[21] 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," in *Proc. of the 17th IEEE Symposium on Reliable Distributed Systems*, 1998, pp. 330-335.
[22] F. Xhafa and A. Abraham, "Meta-heuristics for Grid Scheduling Problems," In Meta-heuristics for Scheduling in Distributed Computing Environments, Springer, vol. 146, 2008, pp. 1-37.

Marjan Kuchaki Rafsanjani received her Ph.D. in Computer Engineering, Iran in 2009. She is currently assistant professor at the Department of Computer Science in Shahid Bahonar University of Kerman, Iran. She published about 80 research papers in international journals and conference proceedings. Her main areas of interest are Computer Networks (Wireless Networks, Mobile Ad hoc Networks (MANETs), Wireless Sensor Networks (WSNs)), Electronic Commerce, Artificial Intelligence, Grid & Cloud Computing.

Amid Khatibi Bardsiri received his B.S. degree in computer software engineering from Shahid Bahonar university, Kerman, Iran in 2008, and his M.S. degree in software engineering from Islamic Azad University, Tehran, Iran, in 2010. He is currently undertaking his PhD in science and research branch of Islamic Azad University, focusing on grid computing scheduling. His areas of research include information systems engineering, software development, grid computing, and cloud computing.