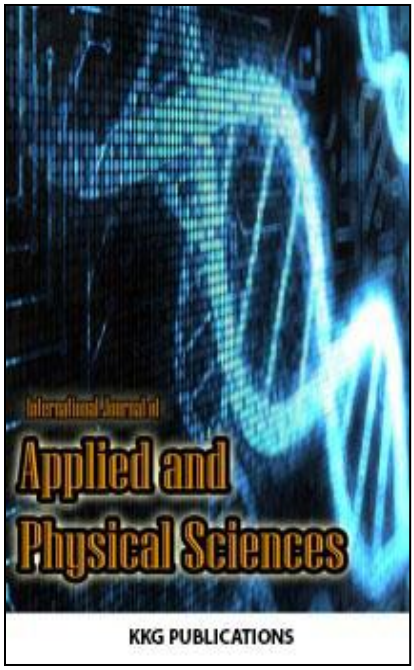

Key Knowledge Generation
Publication details, including instructions for authors and subscription information:
http://kkgpublications.com/applied-sciences/

Range-Suffrage Algorithm for Grid Task

NAGLAA M. REDA[1], A. TAWFIK [2], MOHAMED A. MARZOK [3], SOHEIR M. KHAMIS [4]

[1, 4] Ain Shams University, Cairo, Egypt
[2, 3] Modern University, Cairo, Egypt

Published online: 12 September 2015

To cite this article: N. M. Reda, A. Tawfik, M. A. Marzok and S. M. Khamis, "Range-suffrage algorithm for grid task scheduling," *International Journal of Applied and Physical Sciences*, vol. 1, no. 2, pp. 42-50. 2015.
DOI: https://dx.doi.org/10.20469/ijaps.50004-2
To link to this article: http://kkgpublications.com/wp-content/uploads/2015/12/IJAPS-50004-2.pdf

KKG PUBLICATIONS

International Journal of Applied and Physical Sciences

IJAPS

2015, 2(1): 42-50

RANGE-SUFFRAGE ALGORITHM FOR GRID TASK SCHEDULING

NAGLAA M. REDA [1*] , A. TAWFIK [2], MOHAMED A. MARZOK [3], SOHEIR M. KHAMISA [4]

[1, 4] Ain Shams University, Cairo, Egypt
[2, 3] Modern University, Cairo, Egypt

Index Terms:
Grid Computing
Heuristic Algorithm
Scheduling
Suffrage Algorithm
Resource Utilization
Makespan

Received: 1 April 2015
Accepted: 4 May 2015
Published: 12 September 2015

Abstract. Scheduling is a fundamental process for grid computing systems. Its goal is to map user tasks to suitable resources for execution. The major part of a grid scheduler is to decide which resource is suitable for each task, depending on a scheduling algorithm. Many scheduling algorithms have been designed for reaching optimality. The Suffrage algorithm has shown a superlative performance over most meta-task scheduling algorithms regarding resources selection. However, providing a full power use of resources is still a challenge. In this paper, a new heuristic algorithm is proposed. It aims to maximizing the resource utilization and minimizing the makespan. Its decision is based on detecting the maximum average value of completion times among certain tasks. These tasks are selected depending on their suffrage values. The task having the maximum average is then assigned to the resource with the minimum completion time. Experimental results show that the proposed algorithm outperforms other algorithms in terms of utilization and makespan.

INTRODUCTION

Grid computing has become essential for many scientists, research groups, and standard organizations. Task scheduling is the main step of grid resource management which manages tasks or jobs to allocate resources by using scheduling algorithms and polices. A scheduling algorithm is a part of a scheduler which is responsible for resources discovery, resources selection, job mapping, and job monitoring. These issues are important for efficient execution of a given application on the viable resources. In other words, the scheduler supports strategies for resource discovery and application scheduling, depending on user requirements and manages all issues associated with application execution [1] and [2].

Scheduling algorithms for computational grids may be classified according to a reasonably set having few elements of salient features [1]. The goal of this classification is to provide a commonly accepted set of terminology and to provide a mechanism for allowing comparison of past work in the area of scheduling. In addition, hierarchical classification clarifies the relationships between a scheduling algorithm and another. It is a good guide for future work in this area. Casavant, and Kuhl have presented a hierarchical classification for scheduling algorithms as shown in Figure 1 [3]. The hierarchical divided the scheduling algorithms into two main categories: local scheduling and global scheduling.

The main contribution of this work is to proposing an efficient heuristic for scheduling tasks or jobs to resources on computational grids that maximizes utilization and minimizes makespane. The proposed algorithm (Range-Suffrage) suggested decision for selecting task and resource based on average value of completion time of special tasks. This tasks are selected according to the comparison between the suffrage value and suggested constrains of suffrage range. Performance tests show a competitive improvement over some well-known algorithms.

LITERATURE REVIEW

Local scheduling is responsible for managing a set of heterogeneous resources. Its discipline determines how the tasks resident on a single CPU and executed. On the other hand, global scheduling is responsible for deciding which resources of grid are appropriate to execute a task. A single central authority may do this decision. Global scheduling is divided into two main models which are static and dynamic. Both static and dynamic scheduling focus on when scheduling decision happens. In static scheduling model, information about task execution times and all resources in the grid are available before time application is scheduled. In this model, every task is assigned only once to a specific resource. -But, dynamic scheduling is usually applied when it is difficult to estimate information about task execution times or when jobs arrive on the fly [4] and [5].

Static scheduling methods can be classified into optimal and suboptimal. In optimal scheduling, tasks are allocated to resources depending on some criterion function, e.g., minimum makespan and maximum resources utilization. But recent researchers try to find suboptimal solutions. This suboptimal model can be further divided into two general categories,

*Corresponding author: Naglaa M. Reda
E-mail: r_naglaa2012@yahoo.com

approximate and heuristic. Approximate algorithms do not search for optimal solution. But, heuristic algorithms search for solutions among all visible solutions that are close to the optimal [6]. In this paper, the authors concern with heuristic scheduling algorithms.

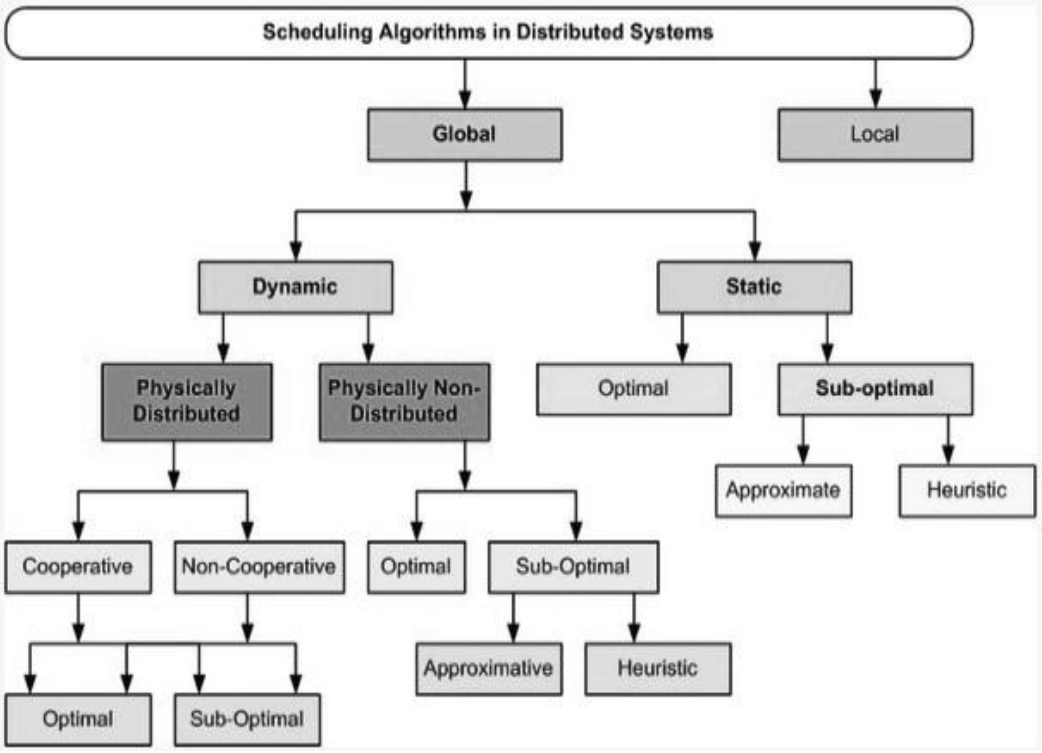

Fig. 1. Taxonomy of scheduling algorithms.

The mapping heuristics can be categorized into two classes: immediate mode and batch mode. In the immediate mode, tasks are mapped onto resources as soon as they arrive. In the batch mode, tasks are collected into a set that is examined for mapping at prescheduled times. The independent set of tasks that is considered for mapping at the mapping events is called a meta-task [7]. There exist many immediate mode algorithms such as Minimum Execution Time MET, Minimum Completion Time MCT, Opportunistic Load Balancing OLB, Switching Algorithm SA and K-Percent Best KPB. Likewise, there exist many batch mode algorithms such as Min-Min, Max-Min, Suffrage, Sort-Mid, and Max-Max [8].

METHODOLOGY

In this section, we designed the new Range-Suffrage algorithm for improving scheduling quality. Our aim is to introduce an efficient heuristic for mapping a set of n tasks to m resources composing a computational grid system G. Our goal is to maximizing the utilization of resources and minimizing the makespan spent to execute all tasks in the given meta-task set. The proposed algorithm is based on the range of suffrage values.

Suppose there exist a grid system $G = \{M_1, M_2, ..., M_m\}$ with m resources; $m>1$, and a finite set of tasks $T = \{T_1, T_2, ..., T_n\}$ containing n independent tasks; $n>1$, that are needed to be executed on the grid G. Then, *Range-Suffrage* scheduling decision is based on a mapping function $S: T \rightarrow G$ which detects for each task T_i the most suitable resources M_j , *i.e.* $S(T_i)=M_j$, where $1< j \leq m$ and $1< i \leq n$.

Range-Suffrage works as follows. First, it initializes the completion time matrix CT. Second, for each task in T, it computes the task suffrage value SV_i is evaluated by different times for task as mentioned in [9]. Then it finds the largest and smallest suffrage values among all tasks which are denoted by LSV and SSV, respectively. Next, it selects those tasks satisfying the range suffrage constrains. This constraint is formulated by the following inequality:

$$LSV \geq SV_i \geq (LSV+SSV)/2.$$

The decision for choosing task is based on computing the average value of completion times of this task on m resources, i.e., $AV_i = \sum_{j=1}^{m} CT_{ij} / m$. Then, a task having the maximum average value is assigned to a resource having minimum completion time to execute this task. After that the assigned task is deleted from T. Finally, the ready time for resource that executed this task is updated. These steps are repeated until all

tasks in T are scheduled in resource. The pseudo code of the algorithm is in the following:

Algorithm Range-Suffrage:

Input: Number of tasks *n*, Number of resources *m*, Tasks $T = \{T_1, T_2, ..., T_n\}$, Ready time of resources *R*; Estimated time of computation *ETC*.
Begin

1. Initialization: $A \leftarrow \{1,2,...,n\}$, $B \leftarrow \{1,2,...,m\}$, $CT \leftarrow ETC$,
2. $resource\_index \leftarrow 0$, $task\_index \leftarrow 0$,
3. $F\_MCT \leftarrow 0$, $S\_MCT \leftarrow 0$, // Denote first and second minimum completion time.
4. $SV \leftarrow 0$, $LSV \leftarrow 0$, $SSV \leftarrow 0$, // Denote maximum and minimum suffrage values.
5. $AV \leftarrow 0$, $M\_AV \leftarrow 0$ // Denote the average and maximum average.
6. While $A \neq \emptyset$ do
7. For all $i \in A$ do
8. For all $j \in$ B do
9. $CT_{i,j} \leftarrow ECT_{i,j} + R_j$;
10. For all $i \in A$ do
11. For all $j \in B$ do
12. Find F_MCT among CT_{ij} and store its resource index in IM_i ;
13. Find S_MCT among CT_{ij} except $CT_{i,IM\ i}$;
14. End For
15. $SV_i \leftarrow S\_MCT - F\_MCT$;
16. Find LSV and SSV among all SV_i : i $\in$ A;
17. $Mean\_Value \leftarrow (LSV+SSV)/2$;
18. For all $i \in A$ do
19. If $SV_i \geq Mean\_Value$
20. Then $AV_i \leftarrow \frac{\sum_{j=1}^{m} CT_{i,j}}{m}$;
21. Else $AV_i \leftarrow 0$;
22. $M\_AV \leftarrow AV_1$;
23. $Resource\_index \leftarrow IM_1$;
24. $Task\_index \leftarrow 1$;
25. For all $i \in A$ do
26. If $M\_AV < AV_i$
27. Then $Resource\_index \leftarrow IM_i$, $Task\_index \leftarrow i$;
28. $S(T_{Task\_Index}) \leftarrow M_{Resource\_index}$;
29. $A \leftarrow A - \{Task\_index\}$;
30. $R_{Resource\_index} \leftarrow R_{Resource\_index} + ECT_{Task\_index,Resource\_index}$;
31. For all $i \in A$ do
32. $CT_{i,Resource\_index} \leftarrow ECT_{i,Resource\_index} + R_{Resource\_index}$;
33. End While
Output: The result of the assignment function *S:* $S(T_1), S(T_2), \ldots, S(T_n)$.

End.

It is clear that Range-Suffrage algorithm is correct, since at the end, the set of tasks indices are vanished, i.e., all tasks are assigned to appropriate resources. In the following, we analyze the time complexity of Range-Suffrage algorithm.

Lemma

The time complexity of Algorithm Range-Suffrage is in $O(n^2 m)$, where n and m are the numbers of tasks and resources in a grid computing system, respectively.

Proof

It is obvious that the first For-loop starting from step 2 to step 4 iterates nm time. For-loop starting from step 5 to step 9 iterates nm time. The other For-loop takes O(n). So, one iteration of while-loop is dominated by O(nm). The while-loop iterates n times. Therefore, the time complexity of the whole algorithm is $O(n^2 m)$□.

RESULTS

For scientists, a computational grid is a complex environment that involves hardware, software, and network, in addition to the unforeseeable behavior of resources. Thus, it is difficult to use a real grid for evaluating or validating their proposals. Therefore, a simulation model is usually used for evaluating new scheduling algorithms. To evaluate the performance of Range-Suffrage proposed scheduling algorithm, we use the benchmark model in [10], [11], [12], [13] and [14]. This benchmark simulation model is commonly used for comparison of static scheduling algorithms for computational environments based on expected time ETC matrix for 512 tasks and 16 resources. It generates twelve different ETC matrices classified in Table 1. They depend upon the three factors: task heterogeneity, resource heterogeneity and consistency. They are labeled as x_yyzz. where x represents a type of consistency (C: consistent, S: semi consistent, I: inconsistent). yy represents the heterogeneity of tasks (hi: high, lo: low). zz represents the heterogeneity of resources (hi: high, lo: low). We use these ETC matrices as input to our computer VB developed program to measure performance of Range-Suffrage against other heuristics.

TABLE 1
TWELVE DIFFERENT ETC MATRICES MODEL

Heterogeneity		Consistency		
Task	Machine	Consistent (c)	Inconsistent (i)	Semi-consistent (s)
High	High	c_hihi	i_hihi	s_hihi
	Low	c_hilo	i_hilo	s_hilo
Low	High	c_lohi	i_lohi	s_lohi
	Low	c_lolo	i_lolo	s_lolo

There are variant performance metrics to evaluate the quality of a scheduling algorithm. We examine two different metrics to validate and evaluate our work. Here, we present results of testing Range-Suffrage algorithm against other heuristic algorithms according to these two different criteria.

Makespan Metric

The makespan is an essential performance criterion of scheduling heuristics for grid scheduler systems. It is the maximum completion time of tasks executed on grid resources. It is computed by using the next equation. Note that CT is the matrix of the completion times after executing given tasks and R is the vector of waiting times of m resources.

$$\text{Makespan} = \max\{ct_{ij} \mid \forall\ 1 \le i \le n, 1 \le j \le m\}, \text{ or}$$
$$\text{Makespan} = \max\{r_j \mid \forall\ 1 \le j \le m\}.$$

TABLE 2
MAKESPAN VALUES OF HIGH TASK, HIGH MACHINE HETEROGENEITY IN CASE OF C, I, AND S BENCHMARK MODELS

Instances / Algorithm	C_hihi	I_hihi	S_hihi
Range-Suffrage	8916741	3178224	4929566
Sort-Mid	9687261	3638838	5487080
Min-Min	8460675	3513919	5160343
Suffrage	10249173	3306819	5121954
SSALB	96784567	3898934	5642247
Max-Min	12385672	8018378	9208811
MET	47472299	4508507	25162058
MCT	11422624	4413583	6693924
OLB	14376662	26102018	19464876

TABLE 3
MAKESPAN VALUES OF HIGH TASK, LOW MACHINE HETEROGENEITY IN CASE OF C, I, AND S BENCHMARK MODELS

Instances / Algorithm	C_hilo	I_hilo	S_hilo
Range-Suffrage	162620	77487	101242
Sort-Mid	171505	87973	113008
Min-Min	161805	80756	104375
Suffrage	168983	77589	102500
SSALB	173288	89151	115498
Max-Min	204055	151924	172823
MET	1185093	96610	605364
MCT	185887	94856	126588
OLB	221052	272785	250362

TABLE 4
MAKESPAN VALUES OF LOW TASK, HIGH MACHINE HETEROGENEITY IN CASE OF C, I, AND S BENCHMARK MODELS

Instances / Algorithm	C_lohi	I_lohi	S_lohi
Range-Suffrage	294006	109134	139393
Sort-Mid	321949	137421	159847
Min-Min	275837	120517	140284
Suffrage	337121	114578	150297
SSALB	312664	135989	162584
Max-Min	392567	251529	282086
MET	1453098	185695	674690
MCT	378304	143816	186151
OLB	477357	833606	603231

TABLE 5
MAKESPAN VALUES OF LOW TASK, LOW MACHINE HETEROGENEITY IN CASE OF C, I, AND S BENCHMARK MODELS

Instances / Algorithm	C_lolo	I_lolo	S_lolo
Range-Suffrage	5520	2706	3620
Sort-Mid	5784	3042	4030
Min-Min	5441	2785	3807
Suffrage	5659	2639	3846
SSALB	5934	3186	4200
Max-Min	6945	5178	6232
MET	39582	3399	21042
MCT	6360	3137	4436
OLB	7307	8938	8938

The makespan of the scheduling algorithms for the twelve different instances of the ETC matrices are shown in Table 2-5. In addition, Table 6 gives the rank of all heuristics based on makespan value of respective schedule for twelve different instances.

TABLE 6
RANK OF HEURISTICS BASED ON MAKESPAN

Rank / Instances	Makespan I	II	III
S_hihi	Range-Suffrage	Suffrage	Min-Min
S_hilo	Range-Suffrage	Suffrage	Min-Min
S_lohi	Range-Suffrage	Min-Min	Suffrage
S_lolo	Range-Suffrage	Min-Min	Suffrage
I_hihi	Range-Suffrage	Suffrage	Min-Min
I_hilo	Range-Suffrage	Suffrage	Min-Min
I_lohi	Range-Suffrage	Suffrage	Min-Min
I_lolo	Suffrage	Range-Suffrage	Min-Min
C_hihi	Min-Min	Range-Suffrage	SSALB
C_hilo	Min-Min	Range-Suffrage	Suffrage
C_lohi	Min-Min	Range-Suffrage	Sort-Mid
C_lolo	Min-Min	Range-Suffrage	Suffrage

Utilization metric

Maximizing the grid's resource utilization of the grid system is important goal for scheduling algorithm [12]. The resource's utilization (RU) is defined as the amount of time at which a resource is not ideal in executing tasks, while the grid's resources utilization GU is the average of RU. They are computed by using the following equations:

$$GU = \frac{\sum_{j=1}^{m} RU_j}{m}, \text{ where } RU_j = \frac{r_j}{makespan}, \text{ for } j = 1,2, \ldots, m.$$

Table 7-9 show the values of GUs for the nine mentioned algorithms. Range-Suffrage gives resource utilization more than 97.9% in I_hihi instance, 98.9 % in I_lohi instance, and more than 99% in other instances. The Max-Min, Sort-Mid, and SSLAB gives the highest maximum resource utilization for some instances but the difference is very small, while the computed makespan of Range-Suffrage algorithm is better than that of Max-Min and SSLAB in all instances.

TABLE 7
GRID'S RESOURCE UTILIZATION (CONSISTENT INSTANCE)

Algorithm / Instances	C_hihi	C_hilo	C_lohi	C_lolo
Range-Suffrage	99.8%	99.9%	99.8%	99.9%
Sort-Mid	99.9%	99.9%	99.9%	99.9%
Min-Min	89.8%	94.8%	88.8%	95.0%
Suffrage	98.0%	99.1%	97.5%	99.2%
SSALB	99.9%	99.9%	99.9%	99.9%
Max-Min	99.9%	99.9%	99.9%	99.9%
MET	6.20%	6.20%	6.20%	6.20%
MCT	95.3%	97.1%	96.9%	95.2%
OLB	94.7%	92.0%	92.9%	92.3%

TABLE 8
GRID'S RESOURCE UTILIZATION (INCONSISTENT INSTANCE)

Algorithm \ Instances	I_hihi	I_hilo	I_lohi	I_lolo
Range-Suffrage	97.9%	99.7%	98,9%	99.3%
Sort-Mid	99.7%	99.8%	99.3%	99.8%
Min-Min	83.4%	91.6%	85.7%	92.0%
Suffrage	92.7%	96.7%	92.6%	98.2%
SSALB	99.6%	99.9%	99.9%	99.9%
Max-Min	99.6%	99.8%	99.9%	99.9%
MET	62.9%	75.1%	53.7%	74.0%
MCT	93.3%	96.0	95.0%	96.6%
OLB	95.1%	95.6%	93.4%	98.0%

TABLE 9
GRID'S RESOURCE UTILIZATION (SEMI CONSISTENT INSTANCE)

Algorithm \ Instances	S_hihi	S_hilo	S_lohi	S_lolo
Range-Suffrage	99.8%	99.9%	99.0%	99.8%
Sort-Mid	99.9%	99.9%	99.7%	99.9%
Min-Min	79.7%	92.4%	88.9%	91.6%
Suffrage	97.4%	99.0%	96.5%	95.9%
SSALB	99.8%	99.9%	99.8%	99.9%
Max-Min	99.8%	99.9%	99.9%	99.9%
MET	11.1%	12.0%	12.2%	12.44%
MCT	92.8%	93.8%	95.4%	95.2%
OLB	96.7%	92.5%	96.2%	95.1%

DISCUSSION AND CONCLUSION

In this paper, we tackle the problem of task scheduling in grid environments. An efficient method called Range-Suffrage to schedule meta-tasks based on range of Suffrage values is proposed. Experiments on twelve different benchmark problems show that Range-Suffrage has effective role on grid scheduling and operates more efficiently than the eight well-known algorithms: Sort-Mid, Min-Min, Suffrage, Simple Scheduling Algorithm with Load Balancing SSALB, Max-Min, MET, MCT, and OLB .

Obviously, results indicate that Range-Suffrage has higher utilization than Suffrage and other algorithms. Range-Suffrage utilizes the grid by 97.9% in I_hihi instance, 98.9 % in I_lohi instance, and more than 99% in other instances. On the other hand, Range-Suffrage has lower makespan than eight algorithms in seven instances. In conclusion, the rank of proposed Range-Suffrage algorithm in both makespan and utilization is superior on most heuristic algorithms.

REFERENCES

[1] T. L. Casavant, and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Trans. Softw. Eng.*, vol. 14, no. 2, pp. 141-154, 1988.

[2] S. Venugopal, R. Buyya and R. Winton, "A grid service broker for scheduling distributed data-oriented applications on global grids," in *Proc. of the 2nd Workshop on Middleware for Grid Computing*, 2004, pp. 75-80.

[3] D. P. Spooner, S. A. Jarvis, J. Cao, S. Saini and G. R. Nudd, "Local grid scheduling techniques using performance prediction," in *IEE Proc. on Conpit. Digit. Tech.*, 2003, vol. 150, no. 2.

[4] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: State of the art and open problems," Rep. of School of Computing held at Queen's Univ. Kingston, Ontario, USA, 2006.

[5] N. Preve, Eds., *"Computational and Data Grids: Principles, Applications and Design: Principles, Applications and Design,"* IGI Global, 2011.

[6] B. A. Shirazi, A. R. Hurson and K. M. Kavi, "Scheduling and load balancing in parallel and distributed systems," IEEE Computer Society Press, 1995.

[7] S. Anousha and M. Ahmadi, "An improved Min-Min task scheduling algorithm in grid computing," in *Grid and Pervasive Computing,"* Springer Berlin Heidelberg, 2013.

[8] N. M. Reda, A. Tawfik, M. A. Marzok, and S. M. Khames, "Sort-Mid tasks scheduling algorithm in grid computing," *J. Advan. Res.*, vol. 6, no. 6, pp. 987-993. 2014.

[9] 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," *J. Parallel Distr. Com.*, vol. 59, no. 2, pp. 107-131, 1999.

[10] G. Sharma, and P. Banga, "Task aware switcher scheduling for batch mode mapping in computational grid environment," *Int. J. of Adv. Res. in Comput. Sci. and Softw. Eng.*, vol. 3, no. 6, pp. 1292-1299, 2013.

[11] H. Izakian, A. Abraham and V. Snasel, "Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments," in *Proc. of the 2009 Int. Joint Conf. on Computational Sciences and Optimization*, 2009, vol. 1, pp. 8-12.

[12] G. Ritchie and J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments," in *Proc. of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, 2004, pp. 1-7.

[13] T. D. Braun, H. J. Siegel and N. A. Beck, "Comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. of Parallel Distr Com, 61*, 810-837, 2001.

[14] M. K. Rafsanjani, and A. K. Bardsiri, "A new heuristic approach for scheduling independent tasks on heterogeneous computing systems," *Int. J. of Machine Learning and Computing*, vol. 2, no. 4, pp. 371-376, 2012.