Scientific Journal of Impact Factor (SJIF):4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 01, January -2018

TDATE: A NEW APPROACH TOWARDS OPTIMAL SCHEDULING IN GRID ENVIRONMENT

P. Muthulakshmi [a]*, Aarthi E [b], P.Yogalakshmi [c]

a,b,c - Department of Computer Science,
SRM Institute of Science and Technology,
Kattankulathur Campus, Chennai, Tamil Nadu, India-603 303

Abstract:- *The major component of any computing system is the scheduling technique that coordinates the entire functionality of system. Heterogeneous Environments depend on scheduling algorithms to execute real life applications. Targeting high performance is found to be the objective of every scheduling algorithm. Scheduling gives the exact plan of how the resources are to be mapped between processes, which turns out as quality performances. The computational demand, diversity among the tasks, heterogeneity of resources, heterogeneity of vendors who offer services, the dynamic nature of the resources are the primary factors in a heterogeneous environment. This paper addresses a scheduling algorithm, which considers dependencies of the tasks, the computation cost and communication cost of the resources to minimize the schedule length of defined sets of tasks on finite number of resources. The algorithm uses bin-packing technique to group the tasks. Tasks from groups are compared and prioritized for scheduling. It is found that the simulated results of the proposed algorithm overcome the compared algorithms. The proposed algorithm outperforms the compared algorithms in terms of schedule length, speed up, load balancing and resource utilization..*

Keywords: *List heuristics, dynamic programming, rank factor, resource finalizing factor, load balance, schedule extent, quick finish time*

1. INTRODUCTION

Scientific, business, and engineering applications are found to be very complex in their architectural and code level. These applications need super computing power in order to be executed. Employing supercomputers are very expensive and it is not easy to maintain these computers. To solve this problem, the idea of orchestrating the available resources that are dedicating themselves across the globe can be adopted. The evolution of distributed computing stayed as great a start to mark the existence of grid technology. Grid technology is the framework that it is accepted as a fully functional environment to solve such problems in an economical way. The grid computing environment is distributed setting that aggregates massive pool of heterogeneous resources which could be utilized by applications. The heterogeneity across the resource type, capacity, speed, location is managed through the model of common and combined configuration set-up, which is common in resource sharing. Another attraction towards grid technology is “reload and execute” from the restore point, when resource failure happens. In order to deploy the application, it must be decomposed into components of codes. Directed Acyclic Graph (DAG) represents the mathematical model of the hierarchical relationships across the modules of the application. The decomposed code components are executed in a parallel and sequential manner based on the dependencies of the components between one another. More dependencies could be a favor for high parallelism. Scheduling algorithms help to execute the components by allocating the components to appropriate grid resources. In general, the process of scheduling involves arranging, managing the resources and work. The message passing interfaces in the scheduling process help to combine the results that are obtained across varied resources. An efficient scheduling is very significant to encounter high performances. A high quality scheduling is meant for its low cost, earlier completion, accurate results; and obviously that might be the key expectation of the clients too. The objective could be reached only when the resources are properly utilized by the applications.

Generally, Directed Acyclic Graphs (DAGs) help us to understand the dependencies and data mobility in the real applications. In this article, we present a scheduling algorithm called Task Dependencies and Affiliate Time Estimation (TDATE) algorithm motivates quick and quality schedules. The algorithm is found to encourage reduced makespan, increases speed up, average grid resources utilization and best rate of result recurrences. The algorithm inherits list heuristics algorithm. The algorithm accomplishes the stages of (i) task selection, (ii) resource selection, (iii) mapping the chosen task and resource. The priority of selection is applied on tasks and resources; task selection is based on population of inter dependent tasks, the resource selection is based on resource availability with respect to start, execution, finish and message passing time. The proposed algorithm does not promote insertion policy as the tasks are encouraged to execute consecutively without idle slots. The algorithm has proven its efficiency in terms of quality of service and grid resource utilization. On

comparisons with the algorithm of its kind, the proposed algorithm arrived at best results when scaled from smaller to bigger task graphs.

Scheduling is an important process in every computing system as its yields performance through proper management of resources. Problem is subdivided into smaller indivisible units called the tasks, which are executed on available resources. Scheduling is a procedure that arrives at decisions to map the tasks and resources. Workflows are structured using task dependencies [1]; in which the communication cost and computation cost are the imperative elements that decide the mapping between tasks and resources. Generally, optimized scheduling algorithms recommend resources of minimum communication cost in terms of transporting links and computation cost in terms of computing resources. We presented a package of existing scheduling algorithms are presented in [2] [3].

Generally, scheduling is categorized into static and dynamic. In static model/deterministic model the information related to the problem is familiar before the scheduling is undertaken. Generally, the scheduling algorithms endorse the performance of a computing system. The task scheduling algorithm is to allocate resources to the tasks and to establish an order for the tasks to be executed by the resources [4]. This paper presents a static scheduling algorithm.

In general, DAG is used to illustrate the workflow of an application. DAG is the composition of nodes and edges, where the nodes represent the tasks and the edges represent communication links. This task system contains a set of tasks with a precedence relation determined only by data dependencies [5]. Tasks are the contestants to use the resources for execution. A task will be bound to the resources until it finishes its execution. The precedence constraint is that the child would start its execution only if all of its parents have completed their execution. This is because the child might need data from its parent(s) to start its execution [6]. In this paper, we propose an algorithm that reduces the makespan and encourages the load balance across resources. The proposed algorithm adapts list scheduling technique and dynamic programming approach [7]. The computation costs and communication costs are considered between nodes/tasks related to three levels (ie. parent, child/children, grand-child/grand-children) for comparison and to decide the task for execution. Also, the dependency of task is exploited for the betterment of schedules.

The continuation of the paper is presented in the following sections in which Section 2 discusses the related works, Section 3 describes the research problem, Section 4 presents the proposed algorithm, Section 5 shows the results and discussions, Section 6 confers the time complexity of TDATE algorithm and Section 7 gives the conclusion and suggestions on future work.

2. RELATED WORKS

A study on scheduling strategies and algorithms is presented in [8]. Static task-scheduling algorithms are classified with one of its major branches headed by heuristic based algorithms, which are further classified into (i) List scheduling algorithms, (ii) Clustering algorithms and (iii) Task Duplication algorithms. List scheduling is based on (i) task priority (ii) resource selection.

Scheduling of any of the above kind is called as DAG scheduling/workflow scheduling. In general, DAG scheduling aims (i) to reduce the schedule length (makespan), (ii) to improve the scheduling efficiency by minimizing communication delay, (iii) to effective load balance among the resources and (iv) to encourage better resource utilization. Task scheduling problem could be depicted as DAG and is expressed as G= (T, E) where, 'G' is the directed acyclic graph and 'T' is the set of tasks (t_i, $1 \leq t_i \leq nt$, nt is the maximum number of tasks) and 'E' denotes the set of edges (inter task dependencies) between tasks($1 \leq e_i \leq ne$, ne is the maximum number of edges)

An edge is the communication link between the tasks exist two different levels. Hierarchy of tasks in the work flow is shown in terms of levels. Tasks in level 'i' can have its child/children in any of the levels greater than its level. No edge can connect tasks belonging to same level. The tasks found between the first level and last-1 level will have child/children. The first level will have tasks of no parents and the last level will have tasks of no child/children.

The COMPCost of t_i, $1 \leq t_i \leq nt$ would be the cost that would be taking to execute itself on a resource. Each t_i must be executed in the same resource until it finishes its execution. Each e_i, $1 \leq e_i \leq ne$ is represented by the COMMCost, the cost taken to transfer data between resources (if parent and child are executed in different resources). The COMPCost would become zero when the parent and child tasks are executed in the same resource. The child task would be waiting to start its execution until all the necessary data from its parent list are available.

Various scheduling strategies that include, algorithm selects the task on the critical path of the DAG ,algorithm aims at the planning of advanced reservation of resources for entire workflow, algorithms that encourage insertion policies are found from works [9] through [16].

An Efficient Dual Objective Grid Workflow Scheduling Algorithm (EDOS) [10], is a list heuristics algorithm that uses rank function, which begins with the phase of advanced reservation of resources for the entire workflow. The algorithm is observed to be static in resource reservation and dynamic in mapping the resources and tasks. It is examined that EDOS is a semi-dynamic algorithm; also the requirement of number of resources is almost equal to the maximum of number of tasks

present the levels of the DAG. Tasks will be waiting unmapped until the resources are available. Existence of parents is checked in all level to find the level-wise computation time which may lead to a maximum probability of unsuccessful execution. Idle time slots get wasted as the algorithm does not employed insertion policy.

3. RESEARCH PROBLEM AND DESCRIPTION

In this section, we define and describe the TDATE scheduling algorithm. The proposed algorithm uses the bind factor for finalizing the task to be executed on a particular resource. The bind factor is derived out of computation costs and communication costs of tasks in three different (not essentially contiguous) levels that have the relationship of parent, child, and grand-child on the execution path, the hierarchical relationship.

TDATE algorithm establishes a triplet relation between the tasks (grand-parent, parent,child) and hence very fast in finalizing the efficient resources for task execution. There is no expectation on resource availability. Avoid idle time slots between the executions of tasks and hence no insertion policy is avoided, which is used in most of the list based scheduling algorithm. Employing insertion policy results in overhead charges. It is found to that the algorithm is scalable as it works for smaller task graphs and even very big tasks graphs of size 3250. The algorithm is executed in the simulation environment used out of GridSim.

The task having more dependencies in the child level and the grand-child level will be given the priority for executing in the resource that minimizes the finish time (FT). The other parallel tasks will be assigned to other available resources based on the earliest finish time. The child having the maximum computation cost would be assigned to the same resource where its parent was executed. The results from other parent tasks which were executed in other resources would be made available to execute the grand-child.

The proposed algorithm starts by calculating Expected COMPutation Cost (ECOMPCost) on resource for each task with respect to speed of each resource.

For i=1 to max_tasks

For j= 1 to max_res

$$ECOMPCost(t_i, p_j) = COMPCost(t_i)/speed(p_j) \quad (1)$$

Then the Average COMPutation Cost (ACOMPCost) is found for each task with respect to number of resources.

$$For\ i = 1\ to\ maxtasks$$

$$ACOMPCost(t_i) = \sum_{j=0}^{Maxprocessors} COMPCOST(t_i, p_j)/maxprocessors \quad (2)$$

Here, the average data transfer rates between resources and average communication start up time are treated as 1.

The communication cost matrix of n x n (n represents number of tasks) size is given the values of communication cost with nodes having direct edges between source and designation. The Direct Communication Cost Matrix of (DCM) size n x n (n represents number of tasks / nodes) is stored with the communication cost of (t_i, t_j), where t_i and t_j are connected by an edge. Therefore, the step length is always one. The Intermediate Communication Cost Matrix (ICCM) of size n x n is stored with the communication cost(s) of (t_i, t_j) where t_i and t_j are connected by intermediate tasks. Here the step length is number of intermediate nodes/tasks plus one. Resources and processors are interchangeably used as the resources mark computational resources

For the tasks at the first level, the Quick Start Time is Zero.

$$QST(t_i, p_j) = 0;\ 1 \le t_i \le nt\ \ and\ \ t_i \in level\ 0, 1 \le p_j \le maxprocessors \quad (3)$$

For the rest, Quick Start Time (QST) and Quick Finish Time (QFT) are recusively calculated as follows.

$$PRT(p_j) = FT(t_p(t_c), p_j)) \quad (4)$$

$$For\ i = 1\ to\ parents(t_c)$$

$$RTP(t_p(i), p_j) = max\{PUT(t_p(i)) + COMMCost(t_p(i), t_c)\} \quad (5)$$

$$QST(t_c, p_j) = max(PRT(p_j), RTP(t_p(i), p_j) \quad (6)$$

$$QFT(t_c, p_j) = ACOMPCost(t_i) + QST(t_c, p_j) \quad (7)$$

where, PRT represents Resource Ready Time,

FT means Finish Time,

RTP symbolizes Reach Time on Resource,

$t_p(t_c)$ signifies Parent task of child task

RUT refers to Resource Utilization Time,

The first task in the first level is executed on the resource that could complete the execution in Minimum COMPutation Cost (MCOMPCost). The following equation finalizes the resource (Resource Finalization Factor (PFF)) for the chosen task,

$$PFF(t_{i=0}) = Processor(min\{ECOMPCost(t_i)\}) \quad (8)$$

For more tasks in the first level, the execution of each task begins with the mapping of tasks of order from t_1 to t_n (tasks in first level) in the ascending order of COMPCost(t_i) on available resources. t_1 is executed on resource p_j with MCOMPCost, t_2 is executed on resource with next MCOMPCost to that of resource p_j, and so on.

For tasks found in second level onwards, the Resources Finalization Factor (PFF) is evaluated as:

$$PFF(t_c(i) = \{COMMCost\left(t_p, t_c(i)\right) + min\{ECOMPCost(t_c(i))\}\};\ \ t_c(i) \in children(t_p) \quad (9)$$

The child having maximum PFF will be executed on the same resource where the parent task was executed and the child having the minimum PFF will be executed in the resource(other than the one which had been given to the child having max(PFF)) where, the COMPCost is comparatively less. Recursively, the next children of either ends (max end and min end) would be choosing their resources to execute. In parallel to the said execution, these tasks will be identifying the child(ren) in the next level (i.e., the grand-children (t_{gc}) of t_p (parents of t_c) with respect to t_c). Search the parent's of t_{gc} and identify the prominent parent of all the t_{gc} whose PFF is comparatively greater than other with respect to other parents. Then, execute the t_{gc} in the same resource by transporting the data from other parents. The other conservative successors (grandchildren) are executed with respect to their MCOMPCost with respect to the PFF.

The schedule length (make span /overall completion time of the DAG) is defined to be the resource's time that completes its work at the last of all the available resources and it must have executed the task in the last level.

4. THE PROPOSED ALGORITHM

Algorithm TDATE(DAG(tasks,weights,edges,edgecost),maxresources)

1. {
2. Compute ECOMPCost(t_i, p_j) of all the tasks,
3. Compute ACOMPCost(t_i) for all tasks
4. Display randomly generated CostMatrix
5. Compute QST
6. Compute QFT
7. Execute tasks of first level in resources of min(QFT)
8. Compute PFF
9. SortAscendingOrder (PFF)
10. While(unassigned tasks exist)
11. {
12. if(max(PFF(children)))then
13. Assign in same resource where parent had completed its execution
14. if(min(PFF(children))then
15. Assign in an another resource where the COMPCost($t_c(i)$) is lesser
16. Compute PFF of grandchildren
17. if(prominent parent)// the task of (max(PFF)) then
18. Assign the task grandchild(ren) where prominent parent was executed and continue execution by obtaining data from other parents
19. else
20. Assign the task grandchild(ren) to resource that could execute in resource having MCOMPCost with respect to PFF
21. next task;
22. }//while ends here
23. Schedule-Extent=max(finish time of all the resources)
24. }//Algorithm ends here

5. RESULTS AND DISCUSSIONS

As a part of this research work, we developed a tool to generate random directed acyclic graphs [17]. The tool could show the pictorial view of the graph with COMMCost and COMPCost, the text based information of the generated graph, database (tables) to store the data. Results of a sample execution are shown in Figure 1 through Figure 4

The inputs to be given to the tool are (i) the total number of tasks (thereof the height factor would be approximated with the maximum number of tasks in each level). DAG's height factor could be altered based on user's data of maximum tasks per level. And not all the levels would be equally distributed, but randomly distributed taking the seed as maximum tasks per level. (ii) Number of resources and their respective speed, further it would b e asking the choice of generating a DAG with either one in the starting and ending level(level '1' and level 'n') or irrespective of the either. (iii) Maximum COMMCost and Maximum COMPCost which would be taken as the seed value to generate the random COMMCost and COMPCost of individual edge and task respectively.

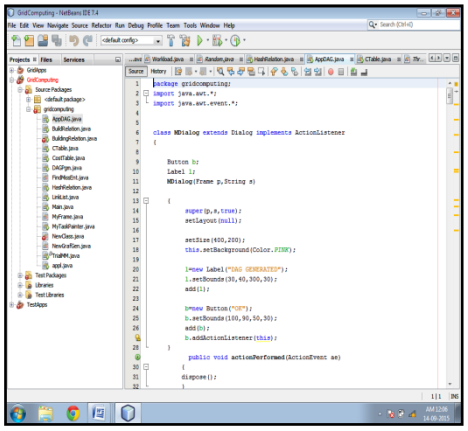

Figure 1. Code to generate DAG

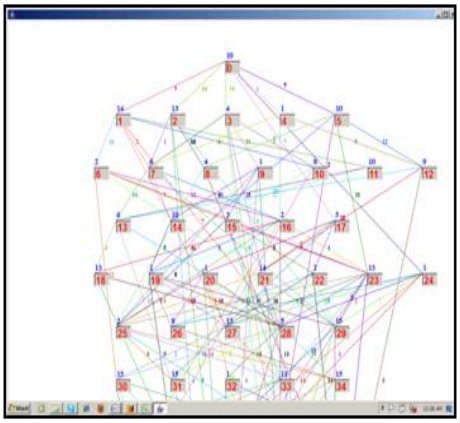

Figure 2. DAG generation

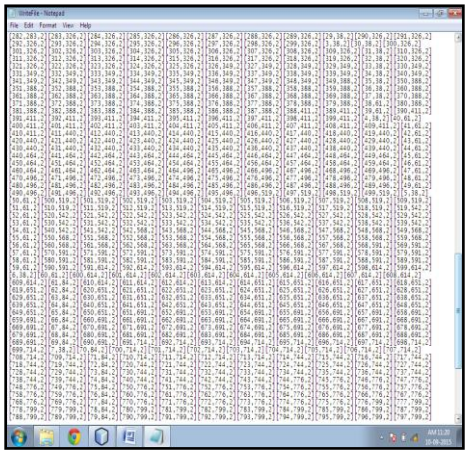

Figure 3. Tasks relationships

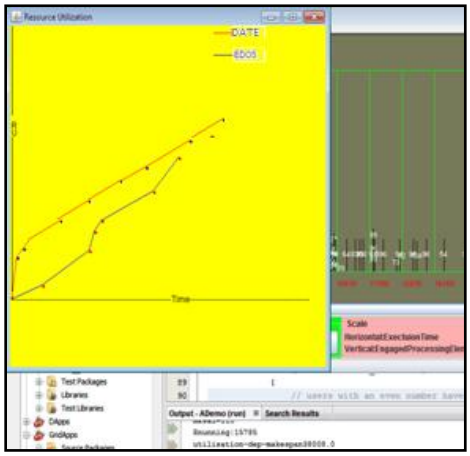

Figure 4. Task Execution(background) and Resource Utilization

The tool helps to calculate the QFT of the tasks (and other factors would be calculated to arrive at QFT). Many combinations of task list and resource list are considered in the simulation process. DAGs of various sizes and characteristics are generated to process the algorithm.

The performance of the algorithms has been scaled are compared with respect to the following attributes,

(i) Schedule length, (ii) Work load between resources
(ii) Execution time of individual tasks, and (iv) Average Resource Utilization

TDATE and EDOS algorithms are experimented with varied set of input parameters of DAG and resources; also it is found that our algorithm gives the best result with respect to schedule extent. As the proposed algorithm totally focuses on

the dependencies of a particular task, it is found that the idle time of a resource between one execution and another is very smaller for big task graph and absolutely zero for small graphs.

Simulation results are tabulated in Table 1 and Table 2 and the comparison outcomes are correspondingly expressed in terms of Gantt Charts in Figure 4 and Figure 5 respectively. To show the results, task graphs/DAG of sizes 50, 75, 100, 125 with different architectures (diamond graph, non-diamond graph) are considered and the resource used to execute the tasks is 4. It is observed that TDATE algorithm outperforms EDOS algorithm in terms of makespan and resource utilization.

TDATE shows a beneficial performance. In the intense graphs, it is observed that the difference between the schedule length of TDATE and EDOS is very big; and in small graphs, the difference is found to be small (and is obvious). It is found that the proposed algorithm produces consistent results on various executions experimented with task graphs of diverging task count on same set of resources.

Table 1. Makespan of TDATE and EDOS algorithms

Tasks	Makespan(msec) (Resources=4)	
	TDATE	EDOS
50	98	106
75	117	129
100	183	199
125	211	226

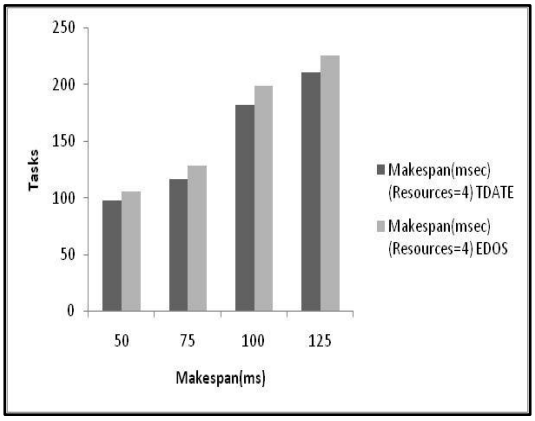

Figure 5. Gantt Chart Makespan

Table 2. Reource Utilization of TDATE and EDOS algorithms

Tasks	Resource utilization(%) (Resources=4)	
	TDATE	EDOS
50	72.05	72.16
75	74.33	71.11
100	79.12	76.66
125	81.09	77.19

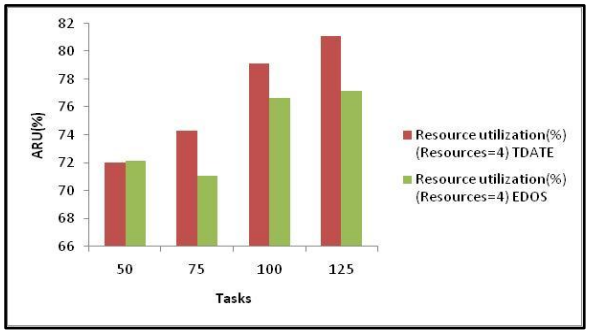

Figure 5. Gantt Chart of Average Resource Utilization

6. TIME COMPLEXITY

The proposed algorithm results in a time complexity of O (2ep), 'e' is the number of edges and 'p' is the count of resources used to execute the tasks. Here, the resource is finalized for the child (thereof for the grandchild) when the parents (parents with respect to child/grandparents with respect to grandchild) are executing themselves and therefore the idle times of the resources are considerably reduced.

7. CONCLUSION

It is observed that the presented algorithm has produced better results on experimenting with various DAGs of different size. It efficiently reduces the schedule length and balancing the load between resource. It is also found that, the rate of best results

is very high among almost all simulations. The proposed algorithm significantly gives an effective speed up than the compared algorithm. The consistency is prevailed on various categories of experiments conducted; and the tested categories are: (i) keeping the same task graph with varied number of resources, (ii) taking different task graphs with same set of resources, (iii) keeping same count of tasks but DAGs are generated with varied relations (a node-id=1(parent in level 1) of a graph will have a set of children(say node-id=2(child 1 in level 2), node-id=3(child in level 2),node-id=7(child in level 3) and node-id=17(child 4 in level 7) and the same node-id=1(parent) of an another graph may not have the same set of children as the graph generated previously) with same set of resources.

The future plan is to improve the algorithm by (i) pertaining task duplication phase, (ii) defining rescheduling point and (iii) to incorporate the algorithm in some real time applications.

REFERENCES

[1] Radu Prodan, Marek Wieczorek, "Bi-Criteria Scheduling of Scientific Work Flows", IEEE Transactions on Automation Science and Engineering. Vol. 7, No. 2, pp. 364-376, 2010.

[2] Y. Kwok and I. Ahamed, "Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiresources", IEEE Transactions on Parallel and Distributed Systems. Vol.7, No. 5, pp. 506-521, 1996.

[3] G. C. Sih and E. A. Lee, "A Compile Time Scheduling Heuristic for Interconnection and constrained Heterogeneous Resource Architecture", IEEE Transaction on Parallel and Distributed Systems, Vol. 4, No. 2, pp. 175-186, 1993.

[4] Haluk Topcuoglu, Salim Hariri, Min-You Wu, " Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3, pp. 260-274, 2002.

[5] Kai Hwang, Faye A. Briggs, "Computer Architecture and Parallel Processing", McGraw-Hill Book Company, International Edition,1985.

[6] Mohammed I Daoud, Nawwaf Kharma, "A Hybrid Heuristic-Genetic Algorithm For Task Scheduling In Heterogeneous Resource Networks", Journal of Parallel and Distributed Computing", Vol. 71, pp. 1518-1531, 2011.

[7] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, "Fundamentals of Computer Algorithms", Galgotia Publications Pvt. Ltd., 2006.

[8] D. I George Amalarethinam, P. Muthulakshmi, "An Overview of Scheduling Algorithms in Grid Computing", International Journal of Research and Reviews in Computer Science, Vol. 2, pp. 280-294, April 2011.

[9] Fang Peng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: state of Art and Open Problems", Technical Report No. 2006-504, School of Computing, Queen's Unversity, Kingston, Ontario, pp. 1-55, 2006.

[10] D.I.George Amalarethinam, F.Kurus Malai Selvi, "An Efficient Dual Objective Grid Workflow Scheduling Algorithm", International Journal of Computer Applications, pp. 7-12, 2011.

[11] Hamid Mohammadi Fard, Hossein Deldari, "An Economic Approach for Scheduling Dependent Tasks in Grid Computing". The 11th IEEE International Conference on Computational Science and Engineering, 2008.

[12] Mandal, A., Kennedy, K., Koelbel,C., Mrin, G., Crummey, J., Lie, B.,johnsson,L., "Scheduling Strategies for Mapping Application Workflows on to the Grid", The 14th IEEE International Symposium on High Performance Distributed Computing, 2005.

[13] Walaa AbdElrouf, Adil Yousif and Mohammed Bakri Bashir, " High Exploitation Genetic Algorithm for Job Scheduling on Grid Computing", International Journal of Grid and Distributed Computing, Vol. 9, pp. 221-228, 2016.

[14] T. Selvakumar et. al., "Implementation of Pervasive Semantic Grid Computing in Hospital Scenario", International Research Journal in Advanced Engineering and Technology, Vol. 2, pp. 517-522, 2016

[15] S. Vaaheedha Kfatheen and Dr. A. Marimuthu, "ETS: An Efficient Task Scheduling Algorithm for Grid Computing", Advances in Computational Sciences and Technology, Vol. 10, pp. 2911-2925, 2017.

[16] Hajara Idris et. al., "An Improved Ant Colony Optimization Algorithm with Fault Tolerance for Job Scheduling in Grid Computing Systems", Plos One, 2017.

[17] D. I George Amalarethinam, P. Muthulakshmi, "DAGITIZER – A Tool to Generate Directed Acyclic Graph through Randomizer to Model Scheduling in Grid Computing", Book: Advances in Intelligence and Soft Computing, Vol. 167, pp. 969-978, 2012.

