An Efficient Dual Objective Grid Workflow Scheduling Algorithm

D.I.George Amalarethinam
Director-MCA & Associate Professor of Computer Science
Jamal Mohamed College, Tiruchirappalli
Tamilnadu, India

F.Kurus Malai Selvi
Assistant Professor of Computer Science
Government College for Women,
Kumbakonam, India,

ABSTRACT - Grid computing is a mainstream technology to integrate large scale distributed sharing resources. To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. Most of the applications in grid computing fall into interdependent task model called workflow application. Task scheduling is a fundamental issue in achieving high performance in grid computing systems. It is well known that the complexity of a general scheduling problem is NP-Complete [1]. The grid workflow task scheduling problem is described by a *Directed Acyclic Graph (DAG)* or task graph. The graph represents the dependency among tasks, their computation time and communication time between them. In the management of workflow execution scheduling, the key issues that impact on the performance of the system is based on proper scheduling. In this paper, a new algorithm, named *Efficient Dual Objective Scheduling (EDOS)* is proposed to maximize the resource utilization in a grid and to minimize makespan by reserving the resources in advance and schedule the task on priority. The proposed algorithm has been implemented for arbitrary task graphs in a simulated environment. Finally, the results are compared with the well known *Min-Min* and *HEFT* scheduling algorithms and showing that the proposed algorithm is yielding better results, that is, minimizing makespan and higher utilization of resources.

Keywords*:* Grid computing, workflow scheduling, inter-dependent tasks, DAG, resource utilization.

1. INTRODUCTION

The Grid connects computers, databases, instruments, and people in a seamless web, supporting computation-rich application concepts such as distributed supercomputing, smart instruments and data-mining. Research on these topics has led to the emergence of a new paradigm known as Grid computing. To achieve the promising potentials of the large number of distributed resources, effective and efficient scheduling algorithms are highly essential. Workflow technology has been used to capture and automate a scientific process that helps scientists to perform their work in grid environment. In order to execute scientific workflow in grid, the task in a workflow needs to be allocated with resources. Scheduling is the problem of deciding the execution time and resource of each of the atomic task [2]. In grid computing, task execution time is depending on the resource to which it is assigned.

Scientific workflows are concerned with the automation processes of interdependent tasks. Workflow management system needs certain functionalities [3] such as

- definition and composition of workflow components,
- task mapping and scheduling during execution and
- data movement between dependent tasks.

The scheduling in the grid can be categorized as full-ahead (static) and just-in-time (dynamic) algorithms. In the static mode every task is assigned once to a resource and its estimated cost of the computation can be made in advance for actual execution. On the other hand, dynamic scheduling is that the system need not be aware of the run-time behavior of the application before.

The workflow task scheduling problem described by a *Directed Acyclic Graph,* called *DAG* scheduling is an optimization problem in the context of traditional homogeneous or heterogeneous parallel computing [4][5][6], but the grid environment is significantly different. Besides the heterogeneity and the possibly substantial communication overheads, there are issues related to the different administration domains that might be involved in providing resources for an application to run. All these issues may hinder the exploitation of parallelism [7]. There are new challenges for scheduling workflow applications in a grid environment are resources sharing on grids, competition for resources, etc [8]. To overcome these challenges the concept of reserving the resources in advance through the *resource brokers* [9]. A *resource broker* is a common gate way to access grid resources.

In the proposed *EDOS* algorithm, the planning of advanced reservation of resources for entire workflow is static but mapping of resources to a particular task is dynamic. In this algorithm the estimation of required resources is easier because the number of tasks in the *DAG* is known in advance. This algorithm has two stages, namely, task preference and resource mapping.

The remainder of this paper is organized as follows. Section 2 describes the related works done; Section 3 specifies the problem statement and defines the terms used in the proposed algorithm. The proposed algorithm and its functional architecture are specified in Section 4. Section 5 compares results of the proposed algorithm with the existing scheduling algorithms. Section 6 provides the conclusion and future enhancement of the work.

2. RELATED WORK

List scheduling is the most commonly used scheduling algorithm by the researchers. An ordered task list is constructed by setting priority for each task in list scheduling. Shoukat Ali et al [10], propose *Min-Min* algorithm which is based on Minimum Completion Time (MCT) of each task with respect to all resources. The task with the overall MCT is selected and assigned to the corresponding resource. Here every task has a good chance to select a suitable resource.

Topcuoglu et al [6] propose *Heterogeneous Earliest Finish Time (HEFT)* algorithm which uses rank function. The HEFT algorithm selects the task with the highest upward rank (an upward rank is defined as the maximum distance from the current task to the existing task, including the computation and communication time) at each step. The rank in HEFT is based

on dependency of tasks, that is, closer to the beginning of a workflow ranked higher and are scheduled first.

List scheduling varies by the priorities assigned to the tasks. Two commonly used task priorities are *b-level* (bottom level) and *t-level* (top level) [4], which are calculated recursively. The *t-level* of a node t_i is the length of the longest path (there can be more than one longest path) from an entry node to t_i (excluding t_i). Here, the length of a path is the sum of all the nodes and edge weights along the path. The *t-level* value corresponds to the earliest starting time of task t_i. The *b-level* of a task t_i is the length of the longest path from t_i to an exit node including the weight of t_i. Kwok and Ahmad [11] state that, scheduling in ascending order of *t-level* tends to account for the topological order of the DAG. The scheduling in descending order of *b-level* tends to prioritize critical path activities. The *t-level* and *b-level* calculations require all the tasks in the workflow.

Dong and Akl [12] present a resource-Performance-Fluctuation-Aware (PFAS) workflow scheduling algorithm for grid computing. It updates task ranks and constructs the critical path dynamically in the scheduling procedure according to the change in performance of available resources. PFAS also adopts a look-ahead approach to assign a critical task. It does not consider the possibility or wrong performance prediction, which is likely in the real situations.

Lee et.al. [13] propose Adaptive Dual Objective Scheduling (ADOS) algorithm as a semi-dynamic heuristic, which statically generates the initial schedule using an evolutionary technique and adapts it dynamically as the performance of resources changes; hence, the algorithm is semi-dynamic. The dual objective functions are makespan and resource usage. It achieves the objective function by combining a static heuristic schedule scheme with a dynamic rescheduling.

Sheng Di and Cho-Li Wang [14] present Dynamic Shortest Makespan First (DSMF) algorithm, which handles the workflow with the shortest remaining makespan at any time and priorities its tasks in the scheduling in Peer to Peer (P2P) grid systems. It shows satisfactory average efficiency under dynamic situations.

Except Min-Min algorithm, all the remaining algorithms given in this section use the rank function as b-level or t-level or both.

3. PROBLEM DEFINITION

Workflow scheduling models may be deterministic or non-deterministic. In a deterministic model, the dependencies of tasks and data are known in advance; whereas in non-deterministic model, they are only known at run time. A parallel program can be represented by a Directed Acyclic Graph (*DAG*) [4] $G = (V, E)$, where V is a set of v nodes and E is a set of e directed edges. A node in the DAG represents a task which in turn is a set of instructions which must be executed sequentially without preemption in the same resource. The weight of a task t_i is called the computation time and is denoted by $w(t_i)$. The edges in the DAG, each of which is denoted by (t_i, t_j), correspond to the communication messages and precedence constraints among the tasks. It is a predecessor of t_i and t_j is a successor of t_i, that is, $t_i < t_j$ iff $e_{ij} \in E$. The weight of an edge is called the communication cost/time of the edge and is denoted by $c(t_i, t_j)$.

A sample *DAG* is given in Figure 1. The source task of an edge is called the parent task while the sink task is called the child task. A task with no parent is called an entry task and a task with no child is called an exit task. A child task can be carried out only to receive all messages of its parent tasks. When a task and its successor tasks are scheduled to the same resource, the communication cost is zero.

In the proposed algorithm two objective functions are considered, namely, maximizing the resource utilization and minimizing the total completion time (makespan) of a job. Formally it can be defined as:

Minimization of makespan:

$$Min\{\max FT(t_i)\}$$

where $FT(t_i)$ is the finish time of task t_i.

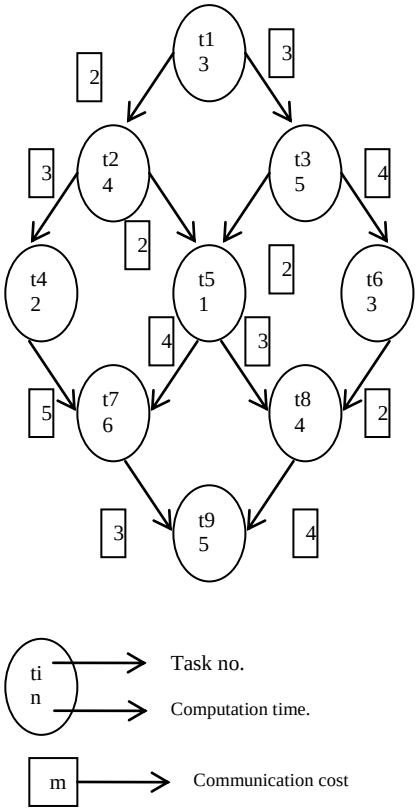

Figure1. A sample DAG

Maximizing the resource utilization of the Grid system is another important objective. The execution time and idle time of a resource are known from the scheduled list is used to calculate the utilization of resources. Resource utilization *(RU(R_i))* of resource R_i is calculated as

$$RU(R_i) = Makespan - \sum_{j=1}^{k} Idletime_j(R_i)$$

where $\sum_{j=1}^{k} Idletime_j(R_i)$ is the sum of all idle time slots of resource R_i.

The average resource utilization *(ARU)* of all resources gives the overall utilization percentage of the grid resources is specified as:

$$ARU = \frac{\left(\sum_{i=1}^{p} RU(R_i)\right)}{p} * 100$$

where p is the total number of resources.

3.1. Terminologies

The proposed algorithm prioritizes the tasks to be scheduled on the basis of a value computed by a *Rank Function (RF).* The *RF* is calculated for each task by level-wise is explicitly specified the i^{th} level in j^{th} task. To compute *RF* the *Ratio of Actual Communication Time (RACT)* and computation time of each task is required.

$$RF(t_{ij}) = RACT(t_{ij}) * w(t_{ij})$$

where *i* refers to number of levels and *j* refers to the number of tasks in a level

$$RACT\left(t_{ij}\right) = \frac{\max(c(pred(t_{ij}), t_{ij}))}{LCT(i)}, \text{ where } t_{ij}$$

is the i^{th} level in j^{th} task,
$max(c(pred(t_{ij}), t_{ij}))$ is maximum communication time between the $pred(t_{ij})$ tasks *and* task t_{ij}, that is, the edge between all parent tasks of t_{ij}. $RACT(t_{ij})$ is calculated for *n* number of tasks, *Level-wise Computation Time LCT(i)* is calculated as follows

$$LCT\left(i\right) = \sum_{k=1}^{l(i)} c(pred\left(t_{ik}\right), t_{ik}), \forall i \in 1,2,..m$$

- number of levels,

where $l(i)$ is the total number of task in a i^{th} level.

The expected computation time (ECT) of task t_{ij} is the actual time required to execute the task in resource r_k is expressed as

$$ECT(t_{ij}, r_k) = w(t_{ij}) / \text{speed of resource}(k).$$

4. AN *EDOS* ALGORITHM

The proposed *EDOS* algorithm has two stages, namely, task preference in static mode and resource mapping in dynamic mode. The problem is represented in the form of *Directed Acyclic Graph (DAG)*. The computation of *RF* is done in task preference stage in static mode because the calculation of *RF* of a task requires computation and communication time from the DAG. The task is assigning to the resource during run time, that is, dynamic mode. This algorithm reserves resources in advance [15]. Advance reservation of resources requires maximum limit of reserving resources. The maximum number of resources required to schedule the task graph is at most the maximum number of tasks in any level of a DAG [6]. After reserving the resources the task preference stage can commence its operation.

In the *EDOS* algorithm two queues are required; the *Ready Task Queue (RTQ)* is used to keep ready the tasks to be given assignment of resources; After their assignment, they may be shifted to another queue, namely, *Finished Task Queue (FTQ).*

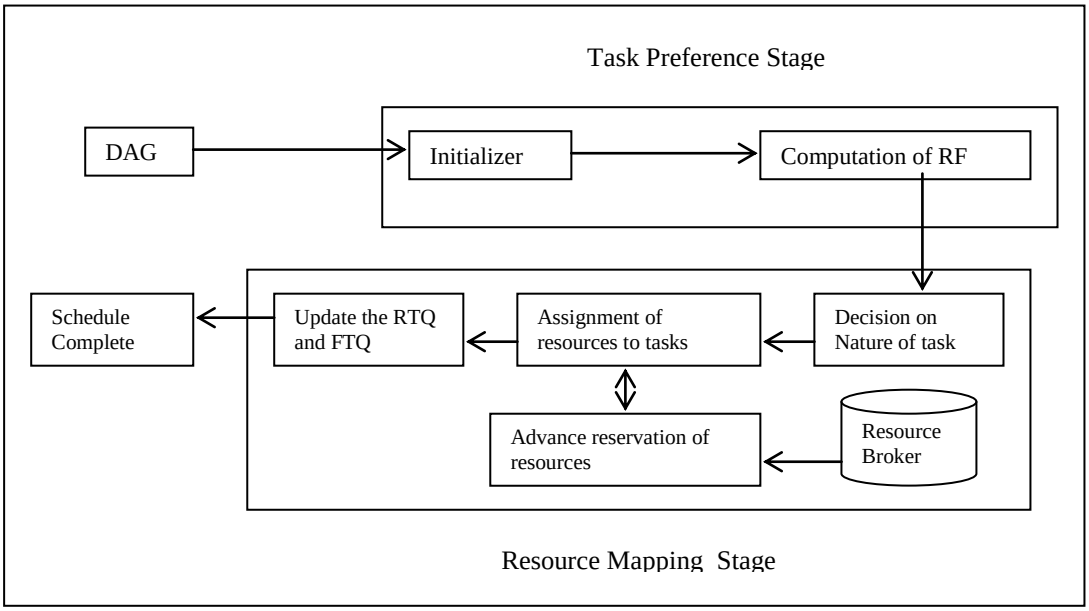

Figure 2.Functional Architecture of *EDOS* Algorithm

4.1. The Algorithm

Procedure ***EDOS***
{
Initialize RTQ and FTQ ={ }, CCT=*0* \\ where CCT is the cumulative communication time
1. Calculate

$$LCT\left(i\right) = \sum_{k=1}^{l(i)} c(pred\left(t_{ik}\right), t_{ik}), \forall i \in 1,2,..m$$

2. Compute

$$RACT\left(t_{ij}\right) = \frac{\max(c(pred(t_{ij}), t_{ij}))}{LCT(i)}, \forall t_{ij} \in n$$

3. Find $RF(t_{ij}) = RACT(t_{ij}) * w(t_{ij})$

4. Reserve *m* number of resources from Grid Resource Broker and place it in Resource
Available List (*RAL*)
5. **While** all tasks are not scheduled do
6. *RTQ* <----- t_{ij}, when parent tasks finished
7. Sort *RTQ* in an ascending order of *RF*
8. **While** (*RTQ*!=empty) do
9. **if** t_{ij} is an entry task or an exit task
10. $r_{k<---}$ *MinECT*(t_{ij}, r_k)
// if the task t_{ij} has more (descendent) nodes to reach end node means schedule it in high speed resources i.e. long path. If it has less number of resources then schedule it in slow speed devices from the *RAL*//
11. **else**
12. **if** *RAL*=even number of resources then
13. t_{ij}<--- maximum $RF(t_{ij})$ task from *RTQ*.

14. **else**
15. take first task t_{ij} from the *RTQ* .i.e. *RF* value is minimum (*RTQ* is already arranged in an ascending order) select resource r_k which has $MinECT(t_{ij},r_k)$
16. schedule t_{ij} in r_k i.e. r_k <--- t_{ij}
17. **endif**
18. **endif**
19. **if** t_{ij} and $pred(t_{ij})$ are scheduled in a same resource then communication time is zero
20. CCT=CCT+0
21. **else**
22. $CCT=CCT+c(pred(t_{ij}), t_{ij})$
23. **endif**
24. Update *RTQ*, *FTQ* and *RAL* //(i.e. remove t_{ij} from *RTQ* and append t_{ij} to *FTQ*, remove resource r_k from *RAL*)//
25. **endwhile**
26. **endwhile**
27. *Makespan = Actual Finish Time (exit task)*
28. $RU(R_i) = Makespan - \sum_{j=1}^{k} Idletime_j(R_i)$
29. $ARU = \frac{\left(\sum_{i=1}^{m} RU(R_i)\right)}{m} * 100$
30. Display makespan, CCT, Resource Utilization time, CCR value of DAG

}

4.2. Functional Architecture of an *EDOS* Algorithm.

The functional architecture of *EDOS* architecture has two major modules, namely, *Task Preference Stage* and *Resource Mapping Stage.* The *Task Preference Stage* has two modules, viz., *Initializer* and *Computation of RF* which are executed sequentially. However, the *Resource Mapping Stage* has three modules, viz., *Decision on nature of task, Assignment of resources to tasks* and finally *Update the RTQ and FTQ* is as shown in Figure 2.

The parallel program modeled as *DAG* is given as input to the *Initializer* module where the Queues, *Ready Task Queue (RTQ)* and *Finished Task Queue (FTQ)* are initialized as empty and *Cumulative Communication Time (CCT)* is set to zero. The available number of resources reserved in advance through *Resource Broker* are placed in Resource *Available List (RAL)*. After these initialization, the *Rank Function (RF)* is calculated for each task by computing LCT and RACT.

Based on *RF*, the decision is made on the nature of the task is found, namely, entry task, exit task or neither of the two. The main function of *Decision on nature of task* module is to select a task from *RTQ* for scheduling based on its nature, i.e., the entry task(s) is scheduled, followed by a normal tasks, and finally the exit task(s).

After selecting the tasks from *RTQ* to schedule, the assignment of resources to them for their execution is done by *Assignment of resources to tasks* module from the *RAL*. After the assignment of resource to a task, the *RTQ* and *FTQ* are updated. The *Resource Mapping Stage* process is repeated till all the tasks are scheduled in *DAG*.

5. RESULTS AND DISCUSSION

For the simulation study the number of tasks in the arbitrary task graphs considered is *20, 25* and *30*. The maximum number of resources is required to reserve in advance at most the maximum number of tasks in any level of a DAG. Therefore the number of resources reserved is *4, 6* and *8* to favor all specified tasks. A resource is a basic computational device or service where tasks, jobs and applications are scheduled, allocated and processed accordingly. Resources have their own characteristics such as CPU characteristics, memory, etc. One of the CPU characteristic is the speed of the resource considered for this simulation. The speed is varied as *1, 1.25, 1.5* and *1.75*. The experiment is conducted for each set of tasks, that is, fixed number of tasks with *4, 6* and *8* resources having different speed with *Min-Min, HEFT* and *EDOS* algorithm separately. The possible combinations of experiments are conducted with fixed number of tasks while varying number of resources and vice-versa. The results are tabulated after conducting the experiments. In grid scheduling the three metrics, namely, *makespan*, *communication cost* and *resource utilization* are considered for comparison.

5.1. Makespan

The main performance measure of a scheduling algorithm is the *makespan* of its schedule. *Makespan* comparison among *Min-Min*, *HEFT* and *EDOS* algorithm are carried out and tabulated the results in Table 1. In all cases the *makespan* of *EDOS* algorithm is minimized. The proposed *EDOS* algorithm gives *makespan* for 20 tasks with 4 resources is 63.12, but the *Min-Min* algorithm and *HEFT* algorithm give 81.26 and 75.867 respectively. The same trend is observed in other cases also.

The tabulated results shown in Table 1 are diagrammatically represented in Figure 3 for 6 resources.

Table 1. *Makespan:* Fixed number of resources with varying number of tasks

Resources	Tasks	Makespan		
		Min-Min	HEFT	EDOS
4	20	81.26	75.867	63.12
	25	90.6	84.4	64.73
	30	95.42	90.238	71.77
6	20	57.1	55.41	53.72
	25	59.62	46.99	45.48
	30	73.58	61.26	53.53
8	20	58.38	52.12	51.21
	25	52.38	47.38	44.03
	30	73.96	62.98	56.68

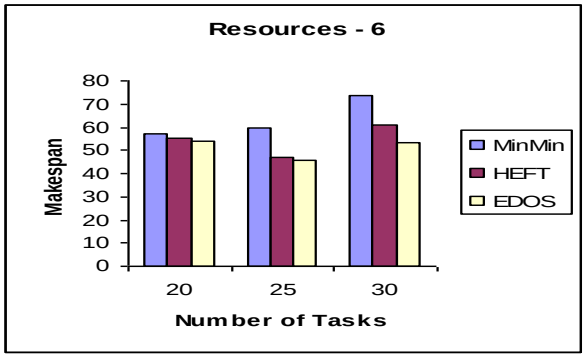

Figure 3. *Makespan* versus number of tasks

5.2. Communication Cost

The communication cost plays a vital role in a scheduling algorithm. The communication cost is high in all cases of *EDOS* algorithm when scheduling with minimum number of resources. As shown in Table 2, the *EDOS* gives communication cost 90 for 20 tasks with 4 resources, but for the same the *Min-Min* and *HEFT* gives 70 and 49 respectively. This tendency is continued in almost all other cases also. The tabulated results shown in Table 2 are diagrammatically represented in Figure 4 for 20 tasks.

Table 2. Communication Cost: Fixed number of tasks with varying number of resources

Tasks	Resources	Communication cost		
		Min-Min	HEFT	EDOS
20	4	70	49	90
	6	65	87	74
	8	68	76	70
25	4	107	88	113
	6	123	140	153
	8	134	149	141
30	4	77	84	121
	6	104	146	146
	8	137	134	133

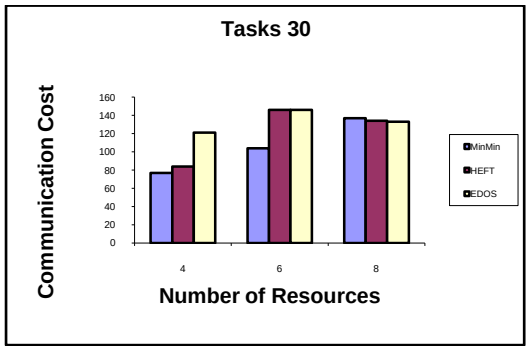

Figure 4. Communication cost versus number of resources

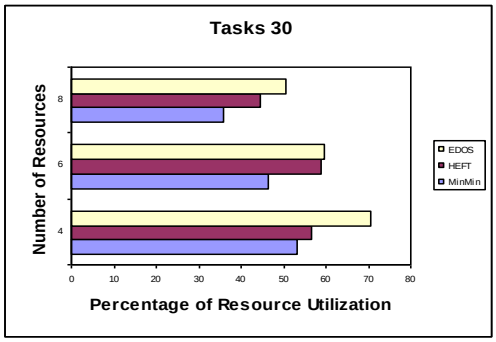

Figure 5. Resource Utilization versus number of resources

The main focusing of *EDOS* algorithm is to minimize *makespan* and maximize resource utilization.

5.3. Resource Utilization

Essentially, advance reservation facilitates to schedule the tasks on various resources. The resources, which have been reserved in advance, are available till the completion of the execution of the last task. As shown in Table 3 the *EDOS* algorithm has given better resource utilization than the *Min-Min* and *HEFT* algorithms in almost all cases. The tabulated results shown in Table 3 are diagrammatically represented in Figure 5 for 30 tasks

Table 3. Resource utilization: Fixed number of tasks with varying number of resources

Tasks	Resources	Resource Utilization %		
		Min-Min	HEFT	EDOS
20	4	42.93	46.31	55.38
	6	33.6	36.7	35.37
	8	26.34	29.1	30.91
25	4	51.27	58.64	69.31
	6	40.54	54.33	54.16
	8	37.7	39.65	44.62
30	4	53.23	56.78	70.52
	6	46.56	58.8	59.52
	8	35.81	44.51	50.55

The Table1, Table 2 and Table 3 confirm that the proposed *EDOS* algorithm minimizes the *makespan* and maximizes *resource utilization* in all the experiments.

6. CONCLUSION AND FUTURE WORK

This paper presents an *EDOS* workflow scheduling algorithm to maximize the resource utilization in a grid and to minimize makespan by reserving the resources in advance. This algorithm integrates the static (task preference stage) and dynamic (resource mapping stage) modes for solving parallel programs which are represented as *DAG*. In *EDOS* algorithm the task with its level are enough to calculate the rank function whereas, the priority based algorithm like *HEFT* requires all tasks from the beginning to the end to calculate the *RF*. The scheduling of this algorithm is not only based on *RF*, but the available number of resources are also considered to utilize them optimally. The tabulated results and its figures have implied that an *EDOS* algorithm fulfilled the dual objective functions.

In future this algorithm is to be focused to reduce the communication cost. Here all the experiments have been conducted with arbitrary task graphs. Real world problems are well balanced and highly parallel are more suitable to schedule using an *EDOS* algorithm. The applicability of this algorithm is to be extended to study in real world workflow applications like Gauss elimination algorithm, Fast Fourier Transformation and a molecular dynamics code. This algorithm does not consider the failure of any task or resource during scheduling and these factors will also be focused in future.

7. REFERENCES

[1] H. El-Rewini, T. Lewis, and H..Ali, Task Scheduling in Parallel and Distributed Systems, ISBN: 0130992356, PTR *Prentice Hall*, 1994

[2] Yves Robert, Frederic Vivien, "Algorithms and Theory of computation Hand Book", *Chapman and Hall CRC*, pp.29.29, November, 2009

[3] María M. López, Elisa Heymann, Miquel A. Senar," Analysis of Dynamic Heuristics for Workflow Scheduling on Grid Systems" *The Fifth International Symposium on Parallel and Distributed Computing(ISPDC'06)*, pp .199-207, July 2006

[4] Y.-K. Kwok and I. Ahmad. Static Scheduling Algorithms for Allocating Directed Task Graphs. *ACM Computing Surveys*, 31(4):pp. 406-471, 1999

[5] R.Sakellariou and H. Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In *Proceedings of 13th Heterogeneous Computing Workshop (HCW 2004)*, Santa Fe, New Mexico, USA ,pp. 26-30, April 2004

[6] H. Topcuoglu, S. Hariri, and M. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):pp. 260–274, March 2002

[7] Henan Zhao and Rizos Sakellariou, "Advance Reservation Policies for Workflows", E. Frachtenberg and U. Schwiegelshohn (Eds.): JSSPP 2006, LNCS 4376, *Springer-Verlag Berlin Heidelberg*, pp. 47–67, 2007

[8] Jia Yu and Rajkumar Buyya, "Workflow Scheduling Algorithms for Grid Computing", *Studies in Computational Intelligence*, Meta-heuristics for scheduling in Distributed Computing Environments, Vol.146,pp. 173-214, 2008

[9]. W. Smith, I. Foster, and V.Taylor.Scheduling with Advanced Reservations. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 127-132, May 2000

[10] Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen and Richard.F.Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", *Heterogeneous Computing Workshop(HCW'99),* pp. 30-44, 1999

[11] Y.-K. Kwok and I. Ahmad, "Benchmarking and comparison of the task graph scheduling algorithms, " *Journal of Parallel and Distributed Computing*, vol. 59, no. 3, pp. 381-422, 1999

[12] Fangpeng Dong and Selim G. Akl, "PFAS: A Resource-Performance-Fluctuation-Aware Workflow Scheduling Algorithm for Grid Computing", *IEEE,* pp.1-9, 2007

[13] Young Choon Lee, Riky Subrata, and Albert Y. Zomaya, "On the Performance of a Dual-Objective Optimization Model for Workflow Applications on Grid Platforms", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 20, No. 9, pp. 1273-1284, 2009

[14] Sheng Di, Cho-Li Wang, "Dual- Phase Just –in –time Workflow Scheduling in P2P Grid Systems",39th *International Conference on Parallel Processing* , pp. 238-247, 2010

[15] Joerg Decker, Joerg Schneider, "Heuristic Scheduling of Grid Workflows Supporting Co-Allocation and Advance Reservation", *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid07)*, pp. 335-342, 2007 .