IJCTA, 9(27), 2016, pp. 227-234 ISSN: 0974-5572

TDSA - Two Directional Dynamic Task Scheduling Algorithm for Heterogeneous Multiprocessor Environment

A. Maria Josphin* and D. I. George Amalarathinam**

ABSTRACT

In multiprocessor environments such as parallel and distributed systems optimizing task scheduling is one of the most important challenges. These systems with task graph mode are modelled by using directed acyclic graph (DAG). The main objective is to minimize the makespan by means of assigning tasks to the processor elements in such a way that precedence constrains are also preserved. This scheduling problem is called NP-complete. In this paper, a new algorithm called TDSA is designed to work with tasks in different levels of DAG. The performance of proposed algorithm TDSA minimizes the makespan, communication to computation ratio (CCR) and maximizes the resource utilization. The parallel scheduling of tasks from different levels yields a better make span, communication to computation ratio (CCR) and resource utilization compared to other existing traditional algorithms.

Keywords: Multiprocessor task scheduling, Parallel and Distributed systems, Task graph.

1. INTRODUCTION

Increase in the usage of multiprocessor systems nowadays in various applications is the result of many breakthroughs over the last two decades. These developments of multiprocessor systems are being used for several applications such as fluid flow, weather modelling, database systems, real-time, and image processing. Data for these applications are evenly distributed on the processors of multiprocessor systems. The maximum benefits of these systems can be obtained by employing an efficient task assignment and scheduling strategy [1]. In this paper, the multiprocessor task scheduling is based on the non-deterministic model, the execution time of tasks and the data communication time between tasks that are assigned; and the directed acyclic task graph (DAG) that represents the precedence relations of the tasks of a parallel processing system [2]. The goal of the scheduler is to assign tasks to available processors such that precedence requirements among tasks are satisfied. Thus, the time required to execute the entire program and the schedule length or makespan is minimized.

Multiprocessor scheduling problems is classified into many different classes based on the following characteristics:

1. The number of tasks and their precedence.
2. Execution time of the tasks and the communication cost, the cost to transmit messages from a task on one processor to a succeeding task on a different processor (Communication cost between two tasks on the same processor is assumed to be zero).
3. Number of processors.
4. Topology of the representative task graph.

DAG represents applications executed within multiprocessor system. In DAG, G = (V, E) consists of a set of vertices V representing the tasks to be executed and a set of directed edges E representing communication dependencies

* Research Scholar, Jamal Mohamed College, Tiruchirappalli, Tamilnadu, India, *Email: sr.mariajose25@gmail.com*

** Director–MCA &Associate Professor of Computer Science, Jamal Mohamed College, Tiruchirappalli, Tamilnadu, India, *Email: di_george@ymail.com*

among tasks. The edge set (E) contains directed edges e_{ij} for each task T_i. The computation weight of a task is represented by the number of CPU clock cycles to execute the task. Given an edge e_{ij}, T_i is called the immediate predecessor of T_j and T_j is called the immediate successor of T_i. An immediate successor T_j depends on its immediate predecessors such that T_j does not execute before it receives results from all of its immediate predecessors. A task without immediate predecessors is called an entry-task and a task without immediate successors is called an exit-task. A DAG may have multiple entry tasks and one exit-task. They are randomly generated models [3,4].

The next section of the paper discusses some studies related to the proposed algorithm, followed by the proposal in details, and section 4 contains the evaluation and results to prove the accuracy of the proposal.

2. RELATED WORK

Several approaches such as heuristic approaches, evolutionary approaches and hybrid methods [6] have been adopted to solve the multiprocessor task scheduling. ReaKook et al [7] presented a comprehensive review and classification of deterministic scheduling algorithms. List scheduling techniques are widely used in task scheduling problems [8]. Insertion Scheduling Heuristic (ISH) and Duplication Scheduling Heuristic (DSH) are well-known list scheduling heuristic methods [9]. ISH is a list scheduling heuristic that was developed to optimize scheduling DAGs with communication delays. ISH extends a basic list scheduling heuristic from GA [10] by attempting to insert ready tasks into existing communication delay slots. DSH [11] improved ISH by using task duplication to reduce the starting time of tasks within a schedule. DSH reduces inter-processor communication time by scheduling tasks redundantly to multiple processors. The genetic-based methods have attracted a lot of researcher attention in solving the Multiprocessor Task Scheduling [12]. One of the main differences in genetic approaches is the usage of genetic operators, such as crossover and mutation. Using different crossover and mutation methods for reproducing the offspring is strongly dependent upon the chromosome representation which may lead to the production of legal or illegal solutions. Another important factor in designing a GA is the simplicity of the algorithm and complexity of the evolutionary optimization process. Moore [13] reported that the results of GA were within 10% of the optimal schedules. The results are based on task graphs with dependencies but without communication delays.

In dynamic scheduling, when a new set of tasks (which correspond to a plan) arrive at the system, the scheduler dynamically determines the feasibility of scheduling these new tasks without jeopardizing the guarantees that have been provided for the previously scheduled tasks. A plan is typically a set of actions that has to be either completed in full or not at all. Each action could correspond to a task and these tasks may have resource requirements, and possibly may have precedence constraints. Thus, for predictable executions, schedulability analysis must be done before a task execution. For schedulability analysis, tasks with worst case computation times must be taken into account. A feasible schedule is generated in which the timing constraints, resource and fault tolerant requirements of all the tasks in the new set can be satisfied. If a feasible schedule cannot be found, the new set of tasks is rejected and the previous schedule remains intact. If the plan is getting to be rejected, the application might invoke an exception task, which must be run, depending on the nature of the plan. This plan allows admission control and results in reservation based system. Tasks are dispatched according to this feasible schedule. Such a type of scheduling approach is called dynamic planning based scheduling [14] and spring kernel is an example for this approach [15].

Abdelkader et al [16] proposed new algorithm Cluster Based Heterogeneous system with Duplication (CBHD) with duplication achieved minimum makespan, maximum utilization and load balancing which are considered as main performance parameters in dynamic environment.

Tabatabae et al [17] proposed linear switching space model using a fuzzy heuristic approach. Fuzzy decision making procedure handles changes in the multiprocessor system while migrating a scheduled task from one processor to another processor to achieve minimum makespan.

Kallia Chronaki et al [18] proposed an algorithm called critically aware dynamic task scheduling on heterogeneous architectures, which shows the potential of a critically aware scheduler to speed up dependency intensive applications

and take advantage of the asymmetric compute resources. It improves the performance by prioritizing the newly created tasks at runtime, detecting the longest path of the dynamic task dependency graph and assigning critical tasks to fast cores.

Vinay Kumar et al [19] introduced an algorithm called Novel Heterogeneous Earliest Finish Time (NHEFT), to enhance the functions of Heterogeneous Earliest Finish Time (HEFT) that works for a bounded number of heterogeneous processors. The main objective of this paper was to achieve high performance and fast scheduling. The proposed algorithm selects the tasks with a rank system and minimizes earliest finish time with minimization cost.

3. PROPOSED ALGORITHM

The process of proposed algorithm starts from the DAG. A DAG contains the parallel tasks called as Parallel Task Queue (PTQ).

The tasks from the PTQ will be assigned to the processors that are taken from the processor bank with the help of bidirectional (TDSA) scheduler.

The proposed scheduler performs two tasks. One is to allot a task to Processors and at the same time manipulate the parent child relationship of tasks. At the initial stage, two directional scheduler creates a CTQ (Child Task Queue) if all the parents of a task are completed. The two directional scheduler schedules the tasks from CTQ in parallel while working with other parent tasks of same level. Once the task is completed, it will be moved to FTQ (Finished Task Queue). The above process will be continued until all the tasks from DAG are allotted to Processors.

Algorithm - TDSA

Step 1: LET D be the input DAG

Step 2: DAG Creates PTQ and PCR Table

Step 3: Allot (Min(T_i), Max(P_j)

Step 4: CT = Fetch Children(T_i)

Step 5: FR = Fetch Free Processor()

Step 6: Allot (ct $i \in CT$, FP_j)

Step 7: do step 3 to 7 until all the tasks allotted

Step 8: Migrate(T_n, Min($p_j \in P$))

Where

PTQ is the Parent Task queue

PCR is the Parent Child Relation Table

T_i is the i^{th} Task

P_j is the j^{th} Processor

CT is the Child Task

FR is the Free Processor

Figure 1 shows the complete architecture diagram of the proposed TDSA scheduling algorithm.

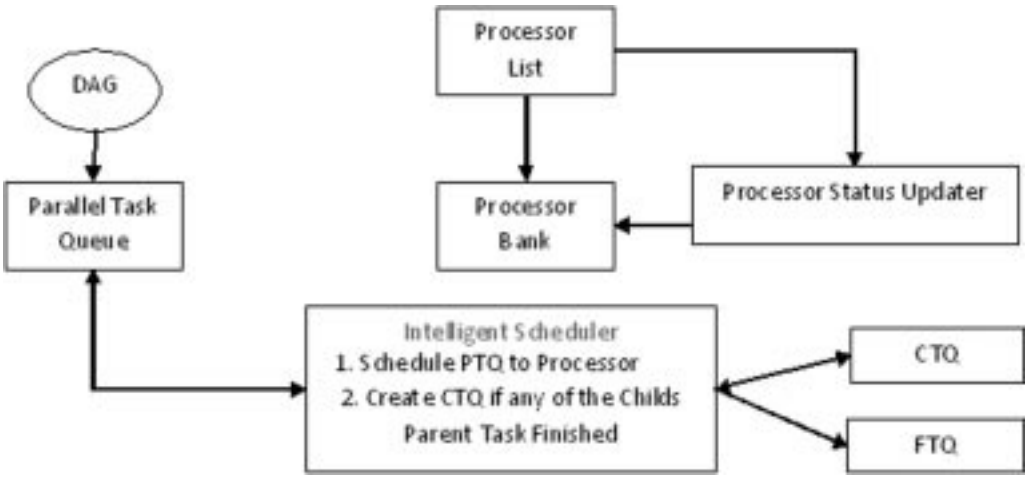


Figure 1: Flow Diagram of Proposed Algorithm

4. EVALUATION AND RESULTS

The main aim of this proposal is to reduce the make span by reducing the waiting time of child tasks for any parent tasks. If T_i is the task having m number of child Tasks. The first child tasks of T_i will be started immediately after the completion of Task T_i.

The waiting time for the completion of all tasks in the same level is eliminated in this proposal. This is to reduce the make span of overall scheduling. But in this proposal the child of any parent task may start immediately after the completion of parent task. The sample DAG with 10 tasks shows in Figure 2. Initial allocation of tasks to the processors is shown in Table 1. Migrated task to the processors is shown in Table 2. Migration is based on the EST (n_i, p_j) and EFT (n_i, p_j), the earliest execution start time and the earliest execution finish time of task n_i on processor p_j respectively.

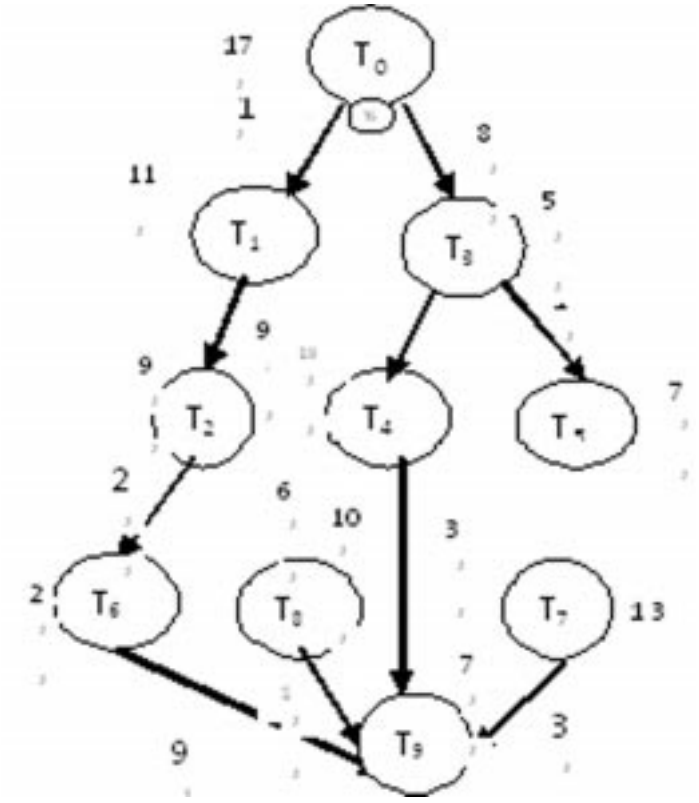


Figure 2: A Sample DAG with 10 Tasks

The EST and EFT values are computed recursively, starting from the entry task as shown in equations (1) and (2). In order to compute the *EFT* of a task all immediate predecessor task n_i of must have been scheduled.

$$EST(n_i,p_j) = \max \quad avil(j, \max nm \notin pred(ni\ AFT(nm + Cm,i \quad (1)$$

$$EFT(n_i, p_j) = w_{i,j}\ EST(n_i, p_j) \quad (2)$$

$$\text{For the entry task } EST(n_{entry}, p_j) = 0 \quad (3)$$

where avail (j) is the same processor p_j is free and it is ready to execute task n_i, the inner max block in equation (1) returns the ready time i.e., the time when all data needed by task has arrived at processor. After all tasks in a graph are scheduled, the scheduled length (overall completion time) will be the *AFT* of the exit task (n_{exit}). If there are multiple exit tasks and the convention of inserting a pseudo exit task is not applied, the schedule length, called makespan, is defined as

$$\text{makespan} = max\ [AFT\ (n_{exit})] \quad (4)$$

Processors have their own characteristics such as CPU characteristics, memory, etc. One of the CPU characteristics is the speed of the processors considered for this result. The speed is varied as 1.25, 1.50 and 1.75. Gantt chart for a given task graph is shown in Figure 3. Figures 4a – 4c shows the comparative study of the makespan with various algorithms.

In TDSA scheduling algorithm there are four metrics such as makespan, communication to computation ratio and resource utilization. These are the performance of parallelism parameter, is considered for comparison.

4.1. Makespan

The main performance measure of a scheduling algorithm is the total execution time of exit task in makespan. Figures 4a – 4c shows that all cases of makespan of proposed algorithm is minimized.

4.2. Communication-to-Computation-Ratio (CCR)

The CCR of a parallel program is defined as the average edge weight divided by the average node weight which can be obtained from the formula.

$$CCR = \frac{Average\ Communication\ Cost}{Average\ Computation\ Cost}$$

When CCR > 1, the DAG is a communication intensive, when the CCR < 1, it is the computation intensive. When the communication cost and the computation cost is equal, i.e., CCR = 1, it is called mixed graph [20,21]. TDSA Algorithm performs better result for communication intensive and computation intensive DAGs. Comparison of DLS algorithm are shown in Figures 5a and 5b.

4.3. Resource Utilization

The processors which have been reserved in advance are available till the execution of the last task. Figures 6a – 6c shows the effective utilization of processors better then DLS.

While designing the scheduling algorithms for efficient parallel processing, the following fundamental aspects achieved: performance, time- complexity, scalability and applicability.

Tasks are distributed to the processors according to its dependency. After completion of parent task, the child task is assigned to the processors which require minimum computation cost.

The dynamic scheduling is implemented using task migration at runtime. Two steps are considered while selection of processor for a particular task:

Table 1
Migration Makespan

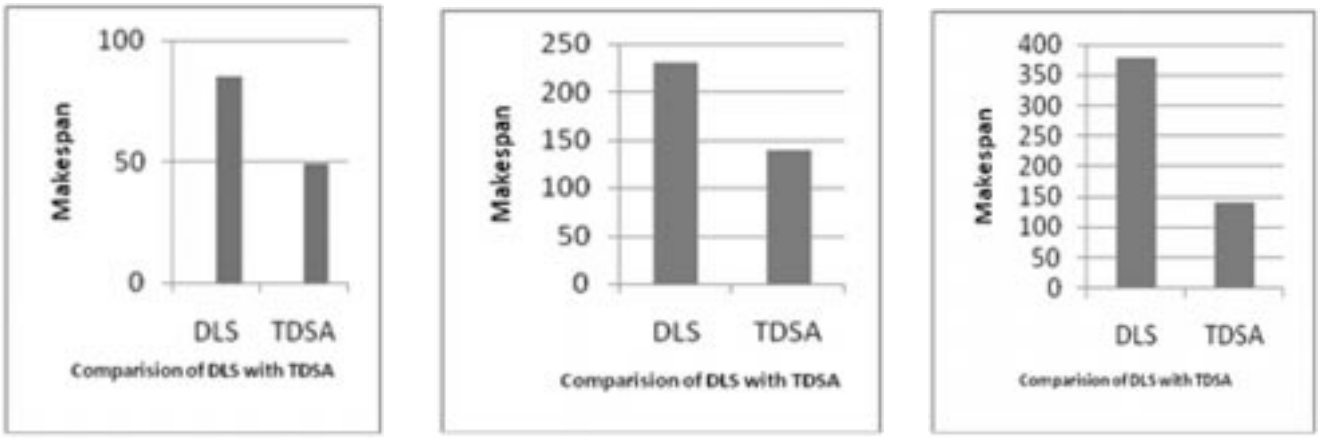
Task	*Processors*	*EST*	*EFT*
T0	P1	0	9.714286
T3	P1	9.714286	12.57143
T1	P0	9.714286	18.04762
T5	P1	12.57143	16.57143
T4	P2	12.57143	28.77143
T2	P0	18.04762	24.04762
T7	P1	16.57143	24
T6	P0	24.04762	25.38095
T8	P2	28.77143	38.77143
T9	P2	38.77143	56.37143
T9	P0	38.77143	49.71429

- Earliest free time of the processor P_j
- Earliest start time of the task T_i on the processors P_j.

If any processor completes its task at the earliest, move the suitable task from the available queue to the processors.

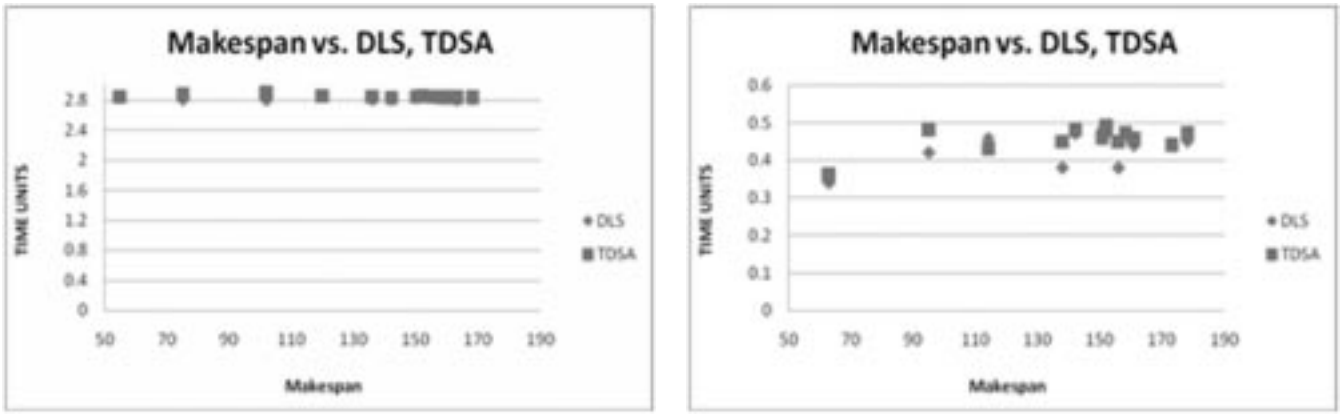
Task/ Processors	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P_0										T1(9-18)								T1(18-24)		
P_1	T0(0-9)									T3(9-12) T5 (12-16)T7 (16-24)										
P_2										T4(12-28)										
Task/ Processors	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
P_0	T1(18-24)																	T9(38-49)		
P_1																				
P_2	T4(12-28)									T8(28-38)										
Task/ Processors	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
P_0	T9(38-49)																			
P_1																				
P_2																				

Figure 3: Gantt chart for the Given Task Graph



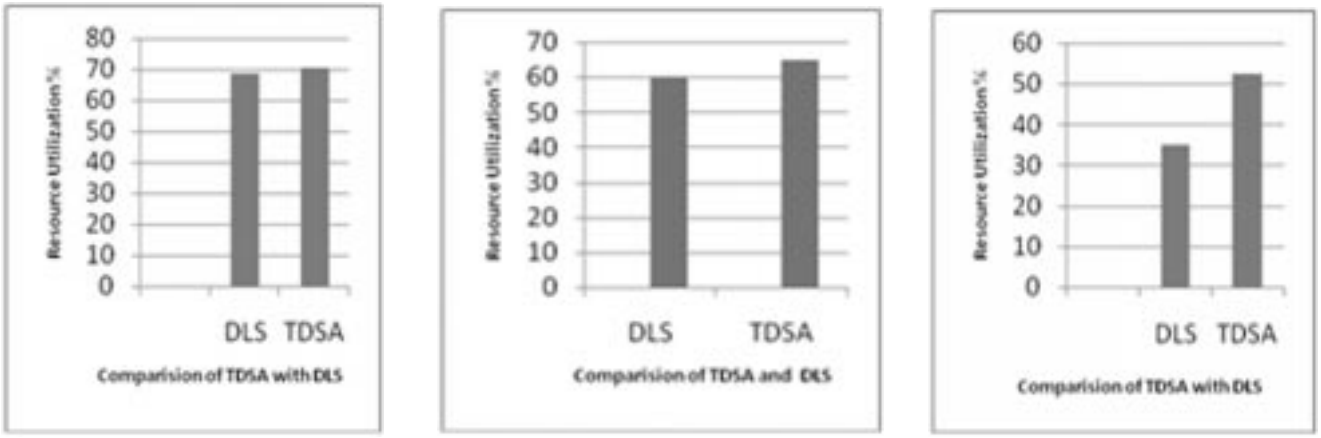
4a. Number of Tasks = 10 4 **b. Number of Tasks = 100** **4c. Number of Tasks = 200**

Figure 4: Comparison of DLS with TDSA algorithm in terms of makespan with different processors



5a. Makespan under Different CCR < 1 5 **b. Makespan under Different CCR > 1**

Figure 5: Comparison of DLS and TDSA algorithm with different CCR values



6a. Number of Tasks = 10 6 **b. Number of Tasks = 100 6** **c. Number of Tasks = 200**

Figure 6: Comparison of DLS with TDSA algorithm in terms of Resource Utilization

5. CONCLUSIONS

In this paper a new DAG based TDSA scheduling algorithm for multiprocessor system is proposed. Scheduling is the decision process by which application components are assigned to the available processors to optimize various performance metrics. The performance of proposed algorithm minimizes the makespan, communication to

computation ratio (CCR) and maximizes the resource utilization. The results of TDSA are compared with many traditional algorithms and it is concluded that the proposed algorithms produced better results for any size of task.

REFERENCES

[1] Garshasbi, M. S. and Mehdi, E., "Tasks Scheduling on Parallel Heterogeneous Multi-processor Systems using Genetic Algorithm", *International Journal of Computer Applications*, **61**, 23-27, 2013.

[2] Hwang, R., Gen, M., Katayama, H., "A comparison of multiprocessor task scheduling algorithms with communication costs", *Computers and Operations Research*, **35**, 976-993, 2008.

[3] Dai, M., Tang C., and Chuang C. "An Energy-Aware Workload Dispatching Simulator for Heterogeneous Clusters", *Proceedings of the International MultiConference of Engineers and Computer Scientists*, **I**, 13 - 15, 2013.

[4] Nissanke, N., Leulseged, A., Chillara, S., "Probabilistic performance analysis in multiprocessor scheduling", *Journal of Computing and Control Engineering*, **13**, 171-179, 2002.

[5] R., RahimiAzghadi,M., Hashemi, S., EbrahimiMoghadam, M., "A hybrid multiprocessor task scheduling method based on immune genetic algorithm" *Qshine 2008 Workshop on Artificial Intelligence in Grid Computing*, 2008.

[6] Wu, A.S., Yu, H., Jin, S., Lin, K.-C., Schiavone, G., "An incremental genetic algorithm approach to multiprocessor scheduling", *IEEE Transactions on Parallel and Distributed Systems*, **15**, 824-834, 2004.

[7] Moore M., "An accurate parallel genetic algorithm to schedule tasks on a cluster", *Parallel and Distributed Systems*, **30**, 567-583, 2004.

[8] Corbalan, J., Martorell, X., Labarta, J., "Performance-driven processor allocation", *IEEE Transactions on Parallel and Distributed Systems*, **16**, 599-611, 2005.

[9] ReaKook H., Mitsuo G. and Hiroshi K., "A Performance Evaluation of Multiprocessor Scheduling with Genetic Algorithm", *ReaKook Hwang et al./Asia Pacific Management Review* , **11**, 67-72, 2006.

[10] Montazeri, F., Salmani-Jelodar, M., Fakhraie, S.N., Fakhraie, S.M., "Evolutionary multiprocessor task scheduling", *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, 2006.

[11] K. Ramamritham and J.A. Stankovic, "Scheduling algorithms and operating systems support for real time systems", *Proc. of IEEE*, **82**, 55-67, 1994.

[12] J.A. Stankovic and K. Ramamritham, The Spring Kernel," A new paradigm for real time operating systems", *ACM SIGOPS Operating Systems Review,***23**, 54-71, 1989.

[13] Doaa M.Abdelkader, Fatma Omara, " Dynamic task scheduling algorithm with load balancing for heterogeneous computing system", *Egyptian Informatics Journal*, **13**, 135-145, 2012.

[14] Hamid TABATABAEE, Mohammad Reza AKBARZADEH-T and Naser PARIZ," Dynamic task scheduling modeling in unstructured heterogeneous multiprocessor systems", *Journal of Zhejiang university – Science (Computers & Electronics)* ,**15**, 423 – 434, 2014.

[15] Kallia Chronaki,Alejandro Rico,Rosa M.Badia Eduard Ayguadé, Jesús Labarta, Mateo Valero,"Critiality –Aware Dynamic Task Scheduling for Heterogeneous Architectures", *Proceedings of the 29th ACM on International Conference on Supercomputing* , 329 – 338, 2015.

[16] Vinay Kumar,C.P.Katti and P.C.Saxena" A Novel Task Scheduling Algorithm for Heterogeneous computing", *International Journal of Computer Applications*, **85**, 2015.

[17] G.J. Joyce Mary and D.I.Geroge Amalarethinam, "A New DAG based Dynamic Task Scheduling Algorithm (DYTAS) for Multiprocessor Systems", *International Journal of Computer Applications*, **19**, 2011.

[18] Hwang RK, Gen M., "Multiprocessor scheduling using genetic algorithm with priority- based coding*", Proceedings of IEEE conference on electronics, information and systems*, 2004.

[19] T. Lucia Agnes Beena and D.I.Geroge Amalarethinam, "Analysis of Task Scheduling Algorithm in Fine-Grained and Coarse – Grained Dags in Cloud Environment", *International Journal of Fuzzy Mathematical Archive*, **6**, 161-168, 2015.

[20] Kamaljit Kaur, Amit Chhabra and Gurvinder Singh, "Modified Genetic Algorithm for Task Scheduling in Homogeneous Parallel System Using Heuristics", *International Journal of Soft Computing*, **5**, 242-251, 2013.

[21] G. Padmavathi and S.R.Vijayalakshmi, "Multiprocessor scheduling for tasks with priority using GA", *International Journal of Computer Science Issues*, **7**, 37-42, 2010.