Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

10th International Conference
on Electrical Engineering
ICEENG 2016

Enhanced Leveled DAG Prioritized Task Scheduling Algorithm in Distributed Computing System

By

Amal EL-NATTAT * Nirmeen A. El-Bahnasawy * Ayman EL-SAYED *

Abstract:

In distributed computing environment, efficient task scheduling is essential to obtain high performance. A vital role of designing and development of task scheduling algorithms is to achieve better makespan. Several task scheduling algorithms have been developed for homogeneous and heterogeneous distributed computing systems. In this paper, we proposed a static task scheduling algorithm that optimizes the performance of Leveled DAG Prioritized Task (LDPT) algorithm; namely ELDPT; to efficiently schedule tasks on homogeneous distributed computing systems. ELDPT algorithm aims to improve the performance of the system by minimizing the schedule length.

Keywords:

Task scheduling; Homogeneous distributed computing systems; Precedence constrained parallel applications; Directed Acyclic Graph.

* Computer Science and Engineering Department, Faculty of Electronic Engineering, Menoufia University

1. Introduction:

Distributed systems have emerged as powerful platforms for executing parallel applications. A distributed system can be defined as a collection of computing systems that appears to its users as a single system, these systems collaborate over a network to achieve a common goal [1]. There are two types of distributed systems; homogeneous (in which processors are identical in capabilities and functionality) and heterogeneous (in which processors are different).

In distributed computing environment, an application is usually decomposed into several independent and/or interdependent sets of cooperating tasks. Dependent tasks are represented by a Directed Acyclic Graph (DAG). DAG can be defined as a graph consists of a set of vertices or nodes and a set of edges G(V, E) in which each node represents a task and each edge represents a communication between two tasks (the two tasks are dependent on each other). The computation cost of the task is represented by a weight associated with each node and the communication cost between two tasks is represented by a weight associated with each edge. The communication cost between two dependent tasks is considered to equal zero if they are executed on the same processor. Figure 1 shows an example of a simple task graph (DAG). In the Figure, t_0 is called predecessor (or parent) of t_2 and t_2 is called successor (or child) of t_0. The edge between t_0 and t_2 means that t_2 can start execution only after t_0 finishes its execution.

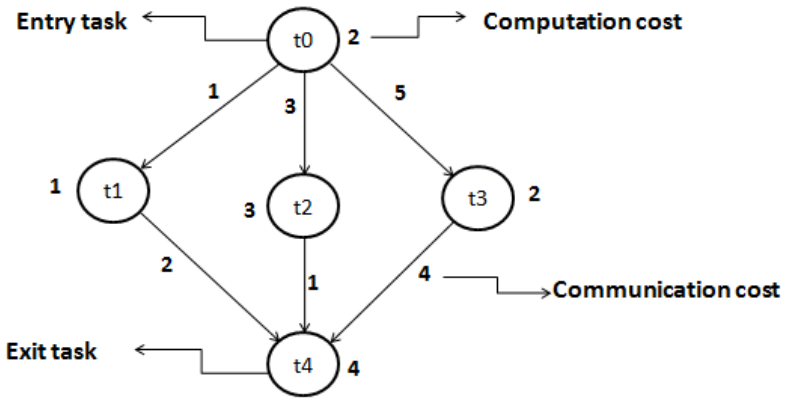


Figure (1): *Example of a DAG*

Efficient task scheduling of application tasks is essential to achieve high performance in parallel and distributed systems. The basic function of task scheduling is to determine the allocation of tasks to processors and their execution order in order to satisfy the precedence requirements and obtain minimum schedule length (or makespan) [2]. Task-scheduling algorithms are broadly classified into two basic classes: static and dynamic.

In static scheduling, the characteristics of an application, such as execution time of tasks and data dependencies between tasks are known in advance (during compile time before running the application). In dynamic scheduling, some information about tasks and their relations may be undeterminable until run-time [3].

Over the past few decades, researchers have focused on designing task scheduling algorithms for homogenous and heterogeneous systems with the objective of reducing the overall execution time of the tasks. Topcuoglu et al. [2] have presented HEFT and CPOP scheduling algorithms for heterogonous processors. Luiz et al. [4] have developed lookahead-HEFT algorithm, which look ahead in the schedule to make scheduling decisions. Eswari, R. and Nickolas, S. [5] have proposed PHTS algorithm to efficiently schedule tasks on the heterogeneous distributed computing systems. Rajak and Ranjit [6] have presented a queue based scheduling algorithm called TSB to schedule tasks on homogeneous parallel multiprocessor system. Ahmed, S.G.; Munir, E.U.; and Nisar, W. [7] have developed genetic algorithm called PEGA that provide low time complexity than standard genetic algorithm (SGA). Xiaoyong Tang; Kenli Li; Renfa Li; and Guiping Liao [8] have presented a list-scheduling algorithm called HEFD for heterogeneous computing systems. Nasri, W. and Nafti, W. [9] have developed a new DAG scheduling algorithm for heterogeneous systems that provide better performance than some well-known existing task scheduling algorithms. In homogeneous distributed systems, researchers have developed many heuristic task-scheduling algorithms such as ISH [10], ETF [11], DLS [12], MH [13], and B-level [14].

In this paper, the problem of scheduling precedence constrained parallel tasks on homogeneous physical machines (PMs) is addressed. The goal of enhanced algorithm is to optimize the performance of LDPT [15] algorithm in order to provide better system performance.

The remainder of this paper is organized as follows. Section 2 provides an overview of LDPT algorithm. The proposed enhanced LDPT algorithm is discussed in section 3. Finally, the conclusion and future work are mentioned in section 4.

2. LDPT Algorithm:

LDPT [15] is a list based scheduling algorithm. It depends on dividing the DAG into levels with considering the dependency conditions among tasks in the DAG. The algorithm has two phases: (1) Task prioritization phase, (2) Processor selection phase. LDPT algorithm depends on giving a priority to each task as shown in Figure 2 then, scheduling each task on one processor with taking into consideration the insertion-based policy. Figure 2 shows the pseudo code of LDPT algorithm.

```
Generate the DAG
Divide the DAG into levels according to their communicated dependency
Sort the constructed levels according to dependency ordering
Sort tasks according to [their computation costs then their direct
communication of its next level] in descending order
While there are unscheduled levels do
    While there are unscheduled tasks do
        For each level do
        Find the task with the highest computation cost
        If there are tasks have equal computation cost
            Then
Choose the task with the highest communication cost with its Childs in next
level
        End if
          Find the processor that minimizes the Earliest Start Time of the
          selected task
          Assign the task to the selected processor
          Remove the selected task from the list
          Repeat
          Until all tasks are scheduled
End for each
End while
```

Figure (2): *LDPT Algorithm [15]*

3. ELDPT Algorithm

New algorithm is a list based scheduling algorithm. It depends on dividing the DAG into levels with considering the dependency conditions among tasks in the DAG. The algorithm has two phases: (1) Task prioritization phase, (2) Processor selection phase.

3.1. Task Prioritization Phase

In this phase, the DAG is divided into levels and the tasks in each level will be sorted into a list in descending order based on their communication costs with children in the next level. The ties are broken by the computation cost of the task.

3.2. Processor Selection Phase

In this phase, the tasks are picked from the list one by one and assigned to the processor that will minimize the earliest start time of the task, with taking into consideration the insertion-based policy. The insertion policy means that if there is an idle time slot on the processor between two already scheduled tasks and it was enough for executing the task, then the task is assigned on that processor in this idle slot without violating precedence constraints. In other words, a task can be scheduled earlier if there is a

period of time between two tasks already scheduled on processor (P), where P runs idle. If two processors provide the same start time for the task then, the task is assigned to the processor on which most of its parents are scheduled. The Earliest Start Time of a task n_i on a processor P_j is defined as shown in equation 1.

$$EST(n_x, P_m) = max[TAvailable(P_m), max\{AFT(n_i) + c_{n,i}\}] \quad (1)$$

Where TAvailable(P_m) is the earliest time at which processor P_m is ready. AFT(n_i) is the Actual Finish Time of a task n_i (the parent of task nx) on the processor P_m. $c_{n,i}$ is the communication cost from task n_i to task n_x, $c_{k,i}$ equal zero if the predecessor task t_k is assigned to processor P_m. For the entry task, EST(n_{entry}, P_m)= 0. Figure 3 shows the pseudo code of the new algorithm.

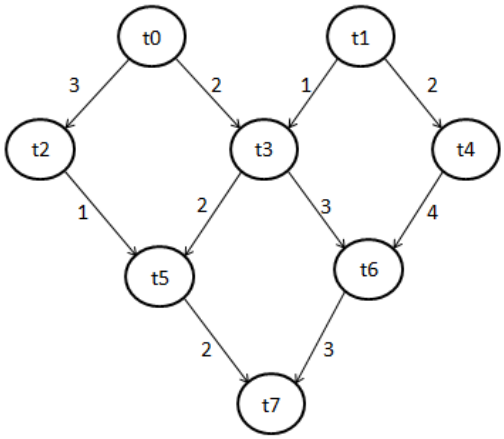
```
Generate the DAG
Divide the DAG into levels according to their communicated dependency
Sort the constructed levels according to dependency ordering
Sort tasks according to [their direct communication of its next level then
their computation costs] in descending order
While there are unscheduled levels do
    While there are unscheduled tasks do
       For each level do
       Find the task with the highest communication cost with its Childs
       in next level
       If there are tasks have equal communication cost
        Then
       Choose the task with the highest computation cost
       End if
         Find the processor that minimizes the Earliest Start Time of the
         selected task
         Assign the task to the selected processor
         Remove the selected task from the list
         Repeat
         Until all tasks are scheduled
End for each
End while
```

***Figure (3):** ELDPT Algorithm*

3.3. Performance Evaluation case studies

Case study 1: Consider the DAG shown in Figure 4; assume the system has two processors (P_0, P_1). Table 1 shows the lists generated by LDPT and ELDPT. Figure 5

shows the Gantt chart generated by LDPT and ELDPT respectively. Both algorithms assign the selected task to the processor that minimizes the start time (EST) of it. For example, in Figure 5(a), the EST for task t_4 on p_0 is 3 and the EST for t_4 on p_1 is 1, so the task t_4 is scheduled on p_1. In Figure 5(b), the same manner if followed with taking into consideration the insertion-based policy. From Figure 5, it is shown that the scheduling length (the finish time of the last task scheduled from the DAG) resulted from LDPT and ELDPT are 15 and 13 unit of time respectively.



Task	Comput. Cost
t_0	2
t_1	1
t_2	3
t_3	2
t_4	4
t_5	5
t_6	2
t_7	1

Figure (4): *Sample DAG of Case study 1*

Table (1): *Task Lists for LDPT and ELDPT, for Case study 1*

Execution	LDPT	ELDPT
1	t_0	t_0
2	t_1	t_1
3	t_4	t_3
4	t_2	t_4
5	t_3	t_2
6	t_5	t_6
7	t_6	t_5
8	t_7	t_7

Figure 5 depicts the Gantt chart generated by LDPT and ELDPT. From the Figure, it is shown that the scheduling length generated from LDPT algorithms is 15 units of time while the scheduling length generated from ELDPT algorithm is 13 units of time. In case of ELDPT, it is observed that there is less periods in which processors are idle than that of LDPT. According to this result, the overall running time of the application will be decreased and the efficiency of the system will be improved

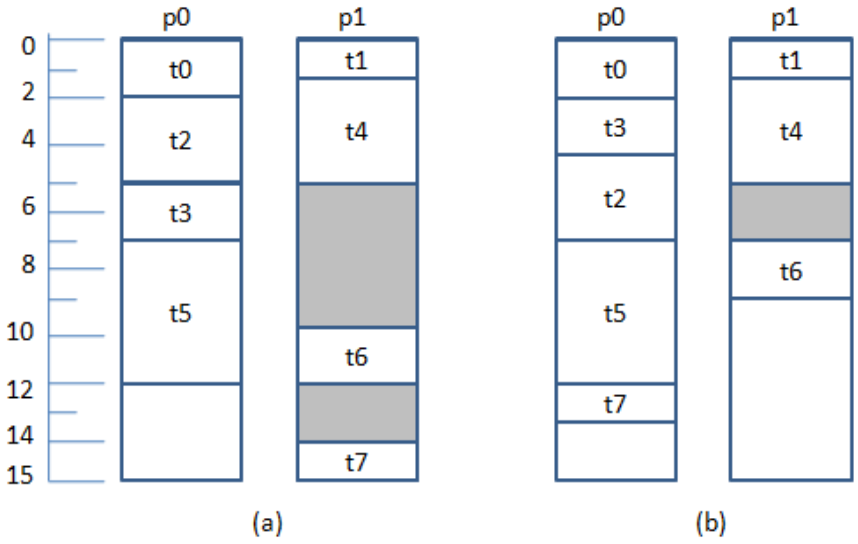
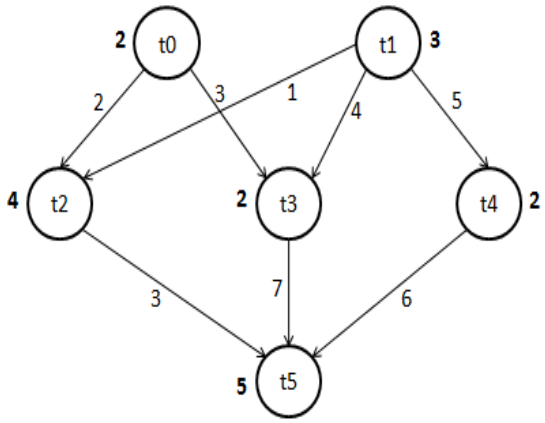


Figure (5): *Case Study 1: Scheduling Length generated by (a) LPDT, (b) ELDPT*

Case study 2: Consider the DAG shown in Figure 6; assume the system has two processors (P_0, P_1). Table 2 shows the lists generated by LDPT and ELDPT. Figure 7 shows the Gantt chart generated by LDPT and ELDPT respectively. In Figure 7(b), t_4 depends on t_1 and its computation cost is 2 units of time so that, t_4 is inserted in the idle time slot between t_1 and t_3 on processor p_0. From Figure 7, it is shown that the scheduling length resulted from LDPT and new algorithm is 21, and 16 unit of time respectively.



Task	Comput. Cost
t0	2
t1	3
t2	4
t3	2
t4	2
t5	5

Figure (6): *Sample DAG of Case Study 2.*

Table (2): *Task Lists for LDPT and ELDPT, for Case study 2*

Execution	LDPT	ELDPT
1	t_1	t_1
2	t_0	t_0
3	t_2	t_3
4	t_3	t_4
5	t_4	t_2
6	t_5	t_5

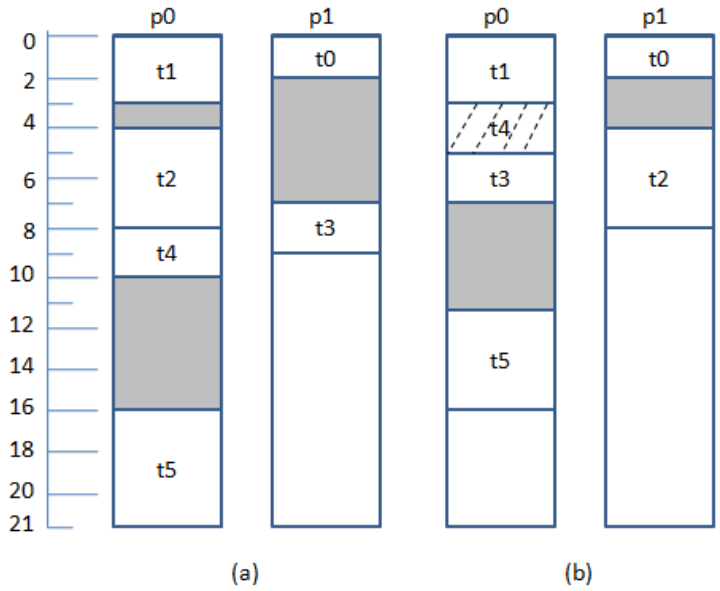
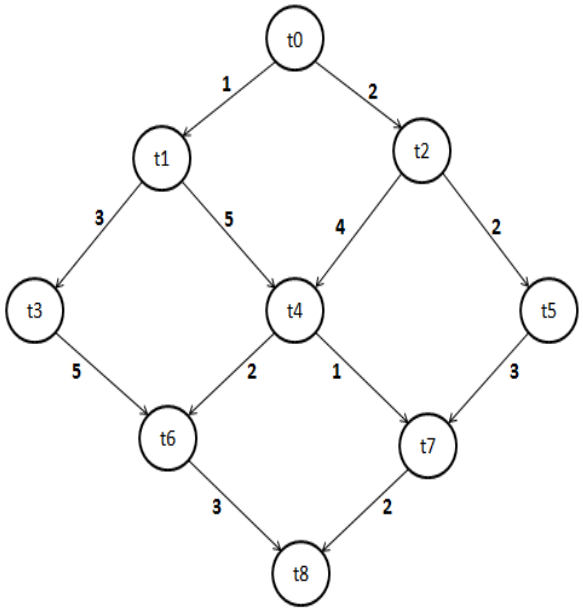


Figure (7): *Case Study 2: Scheduling Length generated by (a) LPDT, (b) ELDPT*

Case study 3: Consider the DAG shown in Figure 8, with the computation cost for each task; assume the system has two processors (P_0, P_1). Table 3 shows the lists generated by LDPT and ELDPT. Figure 9 shows the Gantt chart generated by LDPT and ELDPT respectively. In Figure 9, it is shown that the scheduling length resulted from LDPT and ELDPT are 25, and 23 unit of time respectively.



Task	Comput. Cost
t0	2
t1	3
t2	1
t3	4
t4	3
t5	5
t6	2
t7	4
t8	6

Figure (8): *Sample DAG of Case Study 3.*

Table (3): *Task Lists for LDPT and ELDPT, for Case study 3*

Execution	LDPT	ELDPT
1	t_0	t_0
2	t_1	t_1
3	t_2	t_2
4	t_5	t_3
5	t_3	t_5
6	t_4	t_4
7	t_7	t_6
8	t_6	t_7
9	t_8	t_8

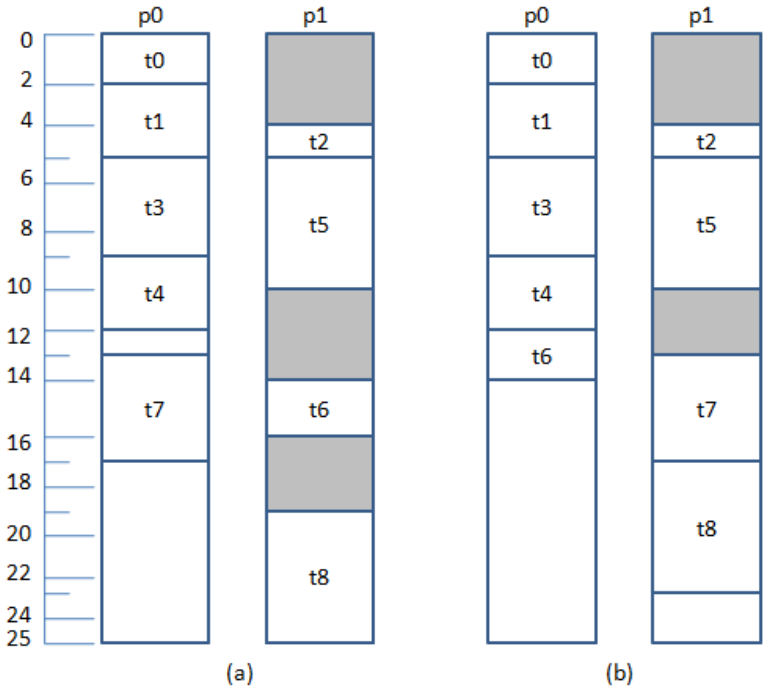
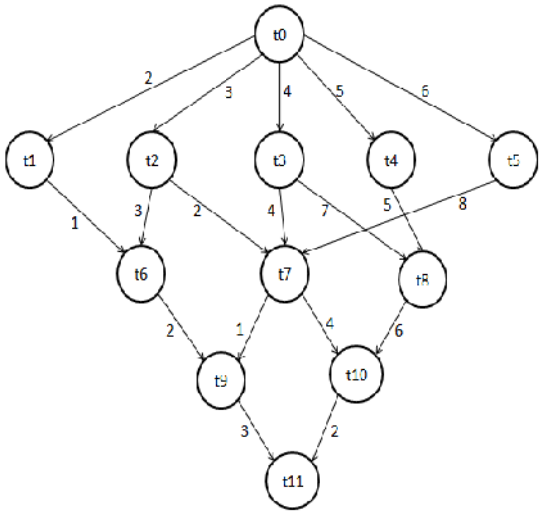


Figure (9): *Case Study 3: Scheduling Length generated by (a) LPDT, (b) ELDPT*

Case study 4: Consider the DAG shown in Figure 10 with the computation cost for each task; assume the system has two processors (P_0, P_1). Table 4 shows the lists generated by LDPT and ELDPT. Figure 11 shows the Gantt chart generated by LDPT and ELDPT respectively. From Figure 11, it is shown that the scheduling length resulted from LDPT and ELDPT are 28, and 26 units of time respectively.



Task	Comput. Cost
t0	3
t1	4
t2	2
t3	1
t4	5
t5	2
t6	6
t7	4
t8	3
t9	2
t10	1
t11	5

Figure (10): *Sample DAG of Case Study 4.*

Table (4): *Task Lists for LDPT and ELDPT, for Case study 4*

Execution	LDPT	New
1	t_0	t_0
2	t_4	t_3
3	t_1	t_5
4	t_5	t_4
5	t_2	t_2
6	t_3	t_1
7	t_6	t_8
8	t_7	t_7
9	t_8	t_6
10	t_9	t_9
11	t_{10}	t_{10}
12	t_{11}	t_{11}

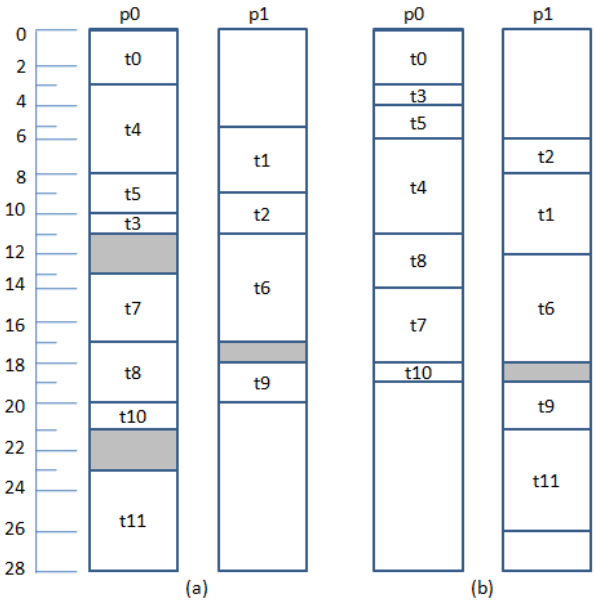


Figure (11): *Case Study 4: Scheduling Length generated by (a) LPDT, (b) ELDPT*

4. Conclusion and Future work

In this paper, a new static scheduling algorithm is developed for homogeneous distributed computing systems. The enhanced ELDPT proposal algorithm is compared with LDPT algorithm theoretically by applying them on some simple DAGs and it provided better scheduling length. In the future this idea will be applied practically by

constructing a simulation environment to compare the performance of LDPT and ELDPT in terms of another evaluation metrics such as schedule length, speed up, and efficiency.

References:

[1] Journal of Theoretical and Applied Information Technology. (2011, April 9). [Online]. Available: http://www.jatit.org/distributed-computing/grid-vs-distributed.htm.

[2] H. Topcuoglu, S. Hariri, and M.Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Systems, Vol. 13, No.3, pp. 260-274, March 2002.

[3] Y.K. Kwok and I. Ahmad, "Static Scheduling Algorithms for allocating Directed Task Graphs to Multiprocessors", ACM Computing Surveys, Vol.31, No.4, pp. 406-471, December 1999.

[4] Luiz F. Bittencourt, Rizos Sakellariou. "DAG Scheduling Using a Look ahead Variant of the Heterogeneous Earliest Finish Time Algorithm", 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing(PDP), pp. 27-34, 2010.

[5] Eswari, R. and Nickolas, S. "Path-Based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems". Advances in Recent Technologies in Communication and Computing (ARTCom), International Conference on 2010. P: 30-34.

[6] Rajak and Ranjit. "A Novel Approach for Task Scheduling in Multiprocessor System". International Journal of Computer Applications (IJCA), Vol.44, No. 11, pp. 12-16, April 2012.

[7] Ahmad, S.G.; Munir, E.U. and Nisar, W. PEGA "A Performance Effective Genetic Algorithm for Task Scheduling in Heterogeneous Systems". High Performance Computing and Communication& 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), IEEE 14th International Conference on 2012. Pp. 1082-1087.

[8] Tang, X., et al., "List scheduling with duplication for heterogeneous computing systems", Journal of Parallel and Distributed Computing (JPDC), Vol. 70, No.4,pp. 323-329.2010.

[9] Nasri,W. and Nafti, W. "A new DAG scheduling algorithm for heterogeneous platforms". Parallel Distributed and Grid Computing (PDGC), second IEEE International Conference on 2012. Pp. 114-119.

[10] B. Kruatrachue and T. Lewis, "Grain size determination for parallel processing," IEEE Software, vol. 5, no. 1, pp. 23-32, May 1988.

[11] J. J. Hwang. Y.C. Chow. F. D. Anger and C.-Y. Lee. "Scheduling precedence graphs In systems with interprocessor communication times." SLAM Journal of Computing, vol. 18, no. 2. pp. 244-257. 1989.

[12] G.C. Slh and E. A. Lee. "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures." IEEE Transactions on Parallel and Distributed Systems, vol. 4. no. 2, pp. 75-87. Feb. 1997.

[13] H. El-Rewini and T.G .Lewis, " Scheduling parallel programs onto arbitrary target machines." Journal of Parallel and Distributed Computing, vol. 9. no. 2, pp. 138-153, June 1990.

[14] Panos M. Pardalos, SanguthevarRajasekaran, José D. P. Rolim, " Randomization Methods in Algorithm Design: DIMACS Workshop", vol. 43, pp. 12-14, December 1997.

[15] Amal EL-Nattat, Nirmeen A. El-Bahnasawy, Ayman EL-Sayed, "A new task scheduling algorithm for maximizing the distributed systems efficiency"; International Journal of Computer Applications, vol.110. no. 9, January 2015.