International Journal of Information Technology and Knowledge Management
January-June 2011, Volume 4, No. 1, pp. 279-284

ANALYSIS, COMPARISON AND PERFORMANCE EVALUATION OF BNP SCHEDULING ALGORITHMS IN PARALLEL PROCESSING

Parneet Kaur[1], Dheerendra Singh[1], Gurvinder Singh[2] & Navneet Singh[3]

This paper surveys algorithms that allocate a parallel program represented by an edge-directed acyclic graph (DAG) based on homogenous processors. The objective is to minimize the execution time, evaluate and compare the performance of the indivuial algorithms and select the best algorithm. Different algorithms are analyzed and classified into four groups. The algorithm in first group schedule the DAG to bounded number of processor (BNP). The algorithm in second group schedule the DAG to unbounded number of processor (UNP).). The algorithm in third group schedule the DAG to task duplication based (TDB). The algorithm in the fourth group perform allocation and mapping on arbitrary processor network topologies (APN). BNP algorithm based upon homogenous environment and it classifies them into four parts (HLFET HIGHEST LEVEL First ESTIMATED time, DLS Dynamic level Scheduling, MCP Modified Critical Path, and EFT Estimated Task First).BNP algorithm is study and discussed, and to measure the performance of BNP algorithm and evaluate the best algorithm.

Keywords: DAG, Multiprocessor, Parallel Processing, Task Graph, List Scheduling

1. Introduction

Parallel processing is the simultaneous processing of the same task on two or more microprocessors in order to obtain the faster results. Given an edge directed acyclic graph (DAG), also called task graph, the problem of scheduling it to a set of homogenous processors to minimize the completion time. DAG is generic model of a parallel program consisting of a set of processes. Each process is an indivisible unit of execution, expressed by node. A node has one or more inputs and can have one or more output to various nodes. The main problem is the scheduling of tasks onto multiple processors and how their performance is analyzed, selecting the parallel processor scheduling algorithm, how to schedule tasks on the processors, how can performance of algorithm be analyzed, on which criteria the performance can be calculated, and selecting the best algorithm. In this paper we try to answer some of these questions by examining a number of recently proposed algorithms. These algorithms can be classified into following categories:

Bounded Number of Processor (BNP) Scheduling [1], [2]: These algorithms schedule the DAG to a bounded number of processor directly. It is based on the list scheduling technique. List scheduling is a class of scheduling heuristics in which the nodes are assigned priorities and placed in a list arranged order of priority.

Unbounded Number of Clusters (UNC) [1],[2] Scheduling: These algorithm schedule the DAG to a bounded number of clusters. The processors are assumed to be fully connected. The technique employed by these algorithms is also called clustering.

Task Duplication Based (TDB) [1] Scheduling: The principle behind the TDB algorithm is to reduce the communication overhead by redundantly allocating some tasks to multiple processors. In duplication different strategies can be employed to select ancestor nodes foe duplication.

Arbitrary Processor Network (AP) Scheduling [1], [2]: The algorithm in these account specific architectural features such as the number of processor as well as their interconnection topology. These algorithms can schedule tasks on the processors and messages on the network communication links.

We discuss the four BNP, UNC, TDB, APN scheduling algorithm. In this paper we study the algorithm using various scenarios and evaluate their relative with respect to each other. The performance parameters are Makespan, Processor Utilization, Speed Up, Critical path and select the best algorithm. The BNP scheduling algorithms are HELFT, MCP, ETF and DLS.

2. DAG Model

The DAG[2],[13],[14] is generic model of a parallel program consisting of a set of processes among which there are dependencies. Each process is an indivisible unit of

[1]Department Computer Science Engineering, Bhai Maha Singh College of Engineering

[2]Department Computer Science Engineering, Guru Nanak Dev University Amritsar

[3]Department of Information Technology, Adesh Institute of Engg. & Tech. Faridkot

Email: [1]dheerendra_singh76@yahoo.co.in

execution, expressed by node. A node has one or more inputs and can have one or more output to various nodes. When all inputs are available, the node is triggered to execute. After its execution, it generates its output. In this model, a set of node {n_1, n_2, n_3.....n_n) are connected by a set of a directed edges, which are represented by (n_i, nj) where n_i is called the Parent node and n_j is called the child node. A node without parent is called an Entry node and a without child node called an Exit node [2]. The weight of a node, denoted by w(n_i), represents the process execution time of a process. Since each edge corresponds to a message transfer from one process to another, the weight of an edge, denoted by c(n_i, n_j),is equal to the message transmission time from node n_i to n_j. Thus, c(n_i, n_j)becomes zero when n_i and n_j are scheduled to the same processor because intraprocessor communication time is negligible compared with the inter processor communication time. The node and edge weights are usually obtained by estimations. Some variations in the generic DAG model are described below:

Accurate Model: In an accurate model, the weight of a node includes the computation time, the time to receive messages before the computation, and the time to send messages after the computation. The weight of an edge is a function of the distance between the source and destination nodes, and therefore, depends on the node allocation and network topology. It also depends on network contention which can be difficult to model. When two nodes are assigned to a single processor, the edge weight becomes zero, so as the message receiving time and sending time.

Approximate Model 1[2]: In this model, the edge weight is approximated by a constant, independent of the message transmission distance and network contention. A completely connected network without contention fits this model.

Approximate Model 2[2]: In this model, the message receiving time and sending time are ignored in addition to approximating the edge weight by a constant.

These approximate models are best suited to the following situations; (i) the grain-size of the process is much larger than the message receiving time and sending time; (ii) communication is handled by some dedicated hardware so that the processor spends insignificant amount of time on communication; (iii) the message transmission time varies little with the message transmission distance, e.g., in a wormhole or circuit switching network; and (iv) the network is not heavily loaded.

In general, the approximate models can be used for medium to large granularity, since the larger the process grain-size, the less the communication, and consequently the network is not heavily loaded. The second reason for using the approximate models is that both the node and edge weights are obtained by estimation, which is hardly accurate. Thus, an accurate model is useless when the weights of nodes and edges are not accurate.

3. BNP ALGORITHM

BNP stands for Bounded Number of processors: These algorithms schedule the DAG to a bounded number of processors directly. The processors are assumed to be fully-connected. Most BNP scheduling algorithms are based on the list scheduling technique[3]. List scheduling is a class of scheduling heuristics in which the nodes are assigned priorities and placed in a list arranged in a descending order of priority. The node with a higher priority will be examined for scheduling before a node with a lower priority. If more than one node has the same priority, ties are broken using some method.

Two major attributes for assigning priority are the t-level (top level) and b-level (bottom level). The t-level of a node n_i is the length of the longest path from an entry node to n_i node in the DAG excluding n_i node. The length of a path is the sum of all the node weights and edge weights along the path. The t-level of n_i is also known as n_i's Earliest start time, denoted by T(n_i), which is determined after n_i is scheduled to a processor.

The b-level of a node n_i is the length of the longest path from node n_i to an exit node. Only the weights of the nodes are considered not weights of the edges while measuring the b-level. The b-level of a node is bounded by length of the critical path. A critical path of a DAG is a path from an entry node to an exit node, whose length is the maximum. The main examples of BNP algorithms are the HLFET (Highest Level First with Estimated Times) algorithm, the MCP (Modified Critical path) algorithm, and the DLS (Dynamic Level Scheduling) algorithm and ETF (Earliest Task First) algorithm [5].

ETF (Earliest task First) Algorithm [1],[6]: Earliest Starting is same as the t-level.

MCP (Modified Critical Path) Algorithm [1],[5] : It uses an attributed called ALAP time of a node as node priority. The ALAP times of the nodes on the CP are just their t-levels.

DL (Dynamic Level) Algorithm [1],[7]: Dynamic level of the node is calculated by subtracting the Earliest Start Time from Static Level.

HELFET: (Highest level first with estimated times) Algorithm [1],[8]: The Algorithm is one of the simplest list scheduling algorithm using Static level as node priority

4. TASK SCHEDULING

The main aim behind the task scheduling problem [9] is to map nodes (tasks) to multiple processors in such way that it requires least time for the completion of all processes, the task dependencies are satisfied and minimum overall scheduling length is achieved. Moreover, it also helps in

attaining parallelism by executing multiple tasks simultaneously.

In this paper four BNP algorithms are studied and their performance is evaluated taking various possible cases. The possible cases are represented by Directed Acyclic Graphs G = (N,E,C,W) where, N is the set of nodes, W is the set of computation costs of the nodes, E is the set of communication edges, C is the set of communication costs of the edges. Also in the order to study these algorithms homogenous computing environment is considered which means processors having same configurations are used for execution. If child task is scheduled on the same processor as on the parent then the cost of communication is not considered and if the child task is scheduled on the different processor as that of the parent, then communication cost is taken into account.

All these algorithm are based on List Scheduling [10] in which the nodes of the graphs are assigned priority and inserted in a list and later on mapped onto the processor according to their respective priority. List scheduling consists of two phases:

Task Prioritizing Phase: In this phase the priority of each node in DAG is computed and assigned.

Processor Selection: Each task is assigned processor with minimum execution time.

The main scheduling attributes [1],[11] used in DAG for assigning priority are as follows. All these priorities are used while evaluating the algorithms.

t-level: t-level of the node n_i in DAG is the length of the longest path from entry node to n_i not including n_i, i.e. the sum of all the nodes computational costs and edges weights along the path.

$$\text{t-level}(n_i) = \max(\text{t-level}(n_m) + w_m + c_{m,i})$$

Where n_m pred (n_i) (predecessors of n_i), w_m stands for computational cost, $c_{i,m}$ stands for the communication cost and t-level $(n_{entry}) = 0$.

b-level: b-level of node n_i in DAG is the length of the longest path from n_i to the exit node, i.e. the sum of all nodes computational costs and edges weights along the path.

$$\text{b-level}(n_i) = w_i + \max(\text{b-level}(n_m) + c_{m,i})$$

where n_m succ(n_i) (successors of n_i), w_m stands for computational cost, $c_{i,m}$ stands for the communication cost and b-level$(n_{exit}) = w(v_{exit})$.

SL (Static Level): If the edges weights are not taken while considering the b-level, then it is called Static Level.

$$SL(n_i) = w_i + \max(SL(n_m))$$

Where n_m succ(n_i) (successors of n_i) and SL $(n_{exit}) = w(v_{exit})$.

CP (Critical Path): It is the length of the longest path from standing node to the exit node in DAG.

EST(Earliest Starting Time) : Earliest Starting Time is same as the t-level.

$$EST(n_i) = \max(EST(n_m) + w_m + c_{m,i})$$

Where $n_m \in$ pred(n_i) (predecessors of n_i), w_m stands for computational cost, $c_{i,m}$ stands for the communication cost and $EST(n_{entry}) = 0$.

LST (Latest Starting Time): Latest Starting Time of node is computed by following the path starting from exit node upwards till the desired node is reached.

$$LST(n_i) = \min(LST(n_m) - c_{m,i}) - w_i$$

Where n_m succ(n_i) (successors of n_i), w_m stands for computational cost, $c_{i,m}$ stands for the communication cost and

$$LST(n_{exit}) = EST(n_{exit}).$$

DL (Dynamic Level): Dynamic level of the node is calculated by subtracting the Earliest Start Time from the Static Level.

5. Performance Evaluation

The performance is the most important factor deciding the algorithm better from each other [11],[12]. Therefore in order to evaluate the performance, some performance metrics are mentioned below:

Makespan: It is defined as the completion time of the algorithm. Lesser the Makespan less time to execute the algorithm more efficient is the algorithm. Makespan is calculated by measuring the finishing time of the exit task by the algorithm.

Processor Utilization: In multiprocessor system, processor work in parallel. There can be some scenario when large amount of work is done by one processor and lesser by others, which the work distribution is not proportionate. Processor utilization measure the percent of time for which the processor performed. It is calculated by dividing the execution times of the tasks scheduled on the processor with the Makespan time of the algorithm.

Processor Utilization (%) = (Total time taken by Scheduled tasks/Makespan)*100.

Speed Up: It is defined as the ratio of time taken by serial algorithm work to the time taken by the algorithm to perform the same work.

Speed Up = Time taken by serial algorithm / Time taken by parallel Algorithm.

Scheduled length Ratio (SLR): It is defined as the ratio of Makespan of the algorithm to Critical Path values of the

DAG. The lesser the values of SLR the more efficient is the algorithm, but The SLR cannot be less than the Critical path Values.

Scheduled length Ratio = Makespan/Critical Path.

6. PERFORMANCE AND COMPARISON

This section comprises of the analytical results of all the algorithms under all three cases i.e. taking into consideration 5 Node Task, 10 Node Tasks, and 15 Node Tasks are determined. The results include the Makespan time of algorithms in seconds, Scheduled Length Ratio and the Processor Utilization by algorithms.

Case1: 5 Task Nodes: From the graphs shown below we can say that results from all the algorithms using 5 Task Nodes the parameters Makespan, SLR, SpeedUp are throughout same for all the algorithms while there is slight variation in processor utilization by ETF algorithm. But rest of the result is mostly the same.

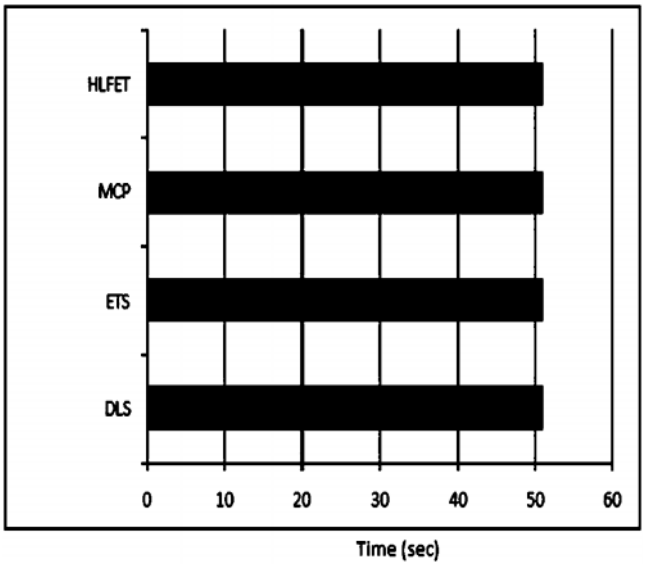

Fig. (i): Makespan for 5 Nodes

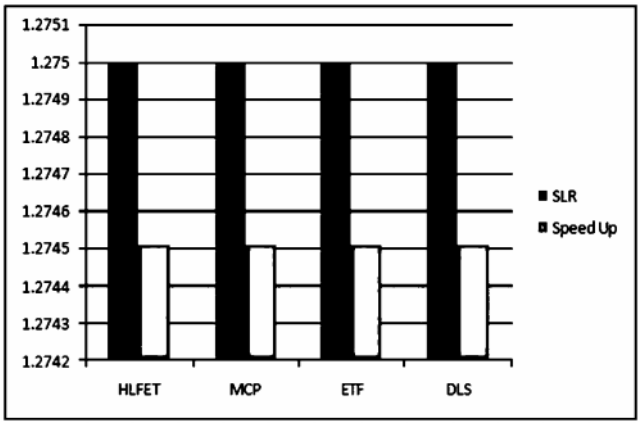

Fig. (ii): SLR and Speedup for 5 Nodes

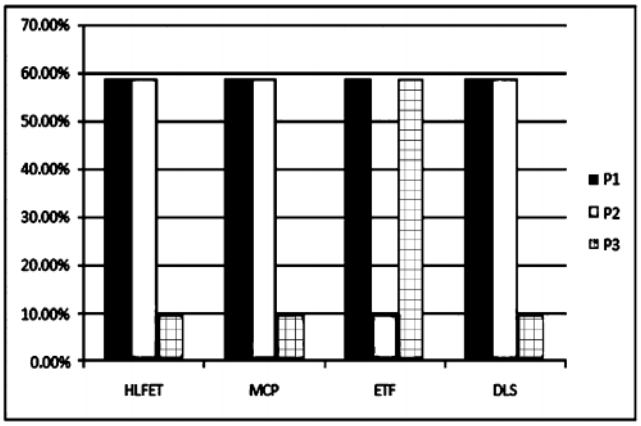

Fig. (iii): Processor Utilization for 5 Nodes

Case 2: 10 Task Nodes: Here the results obtained from all the algorithms using 10 task nodes are different as observed with 5 nodes. The Makespan of MCP, ETF, and DLS is equal, but the Makespan of HLFET is higher than others. Similarly in case with SLR is the same for MCP, ETF, and DLS algorithms but different in case of HLFET.The same applies to SpeedUp and Processor Utilization.

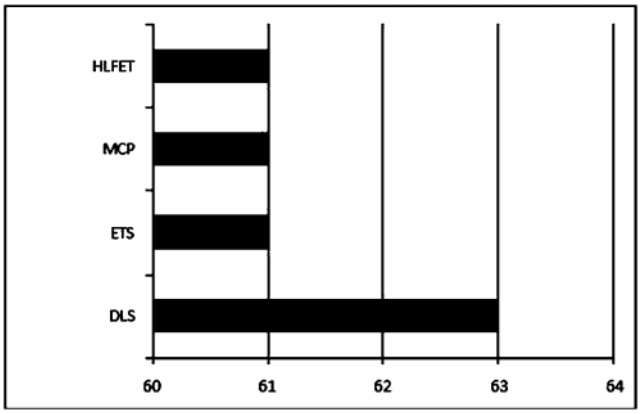

Fig. (iv): Makespan for 10 Nodes

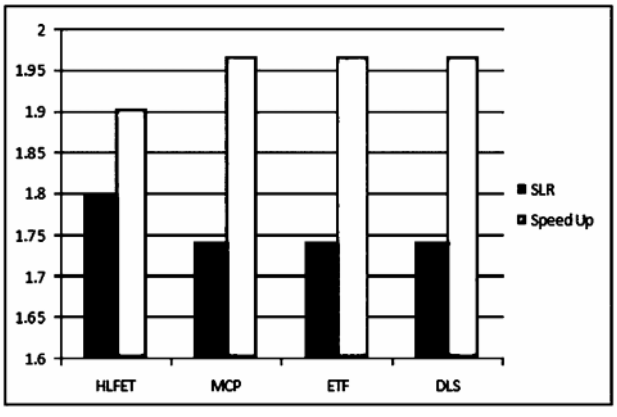

Fig. (v): SLR and Speedup for 10 Nodes

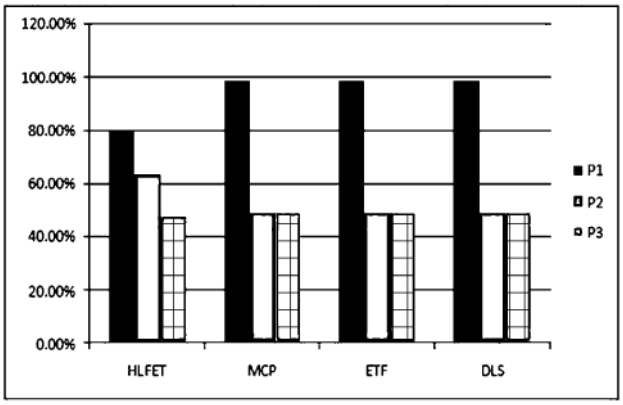

Fig. (vi): Processor Utilization for 10 Nodes

Case 3: 15 Task Nodes: Results obtained for all the algorithms using 15 task nodes are entirely different. The DLS algorithm shows the least Makespan and SLR values with highest SpeedUp values. The ETF algorithm shows the highest Makespan and SLR values with lowest SpeedUp values.

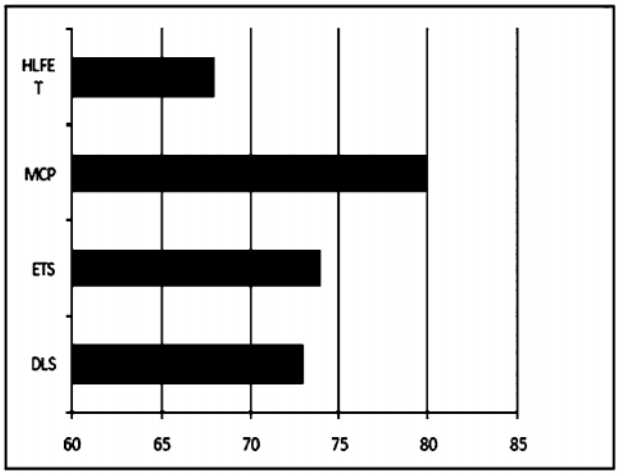

Fig. (vii): Makespan for 15 Nodes

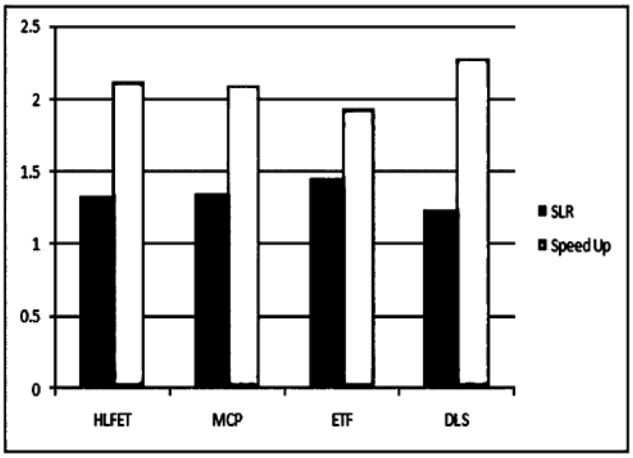

Fig. (viii): SLR and Speedup for 15 Nodes

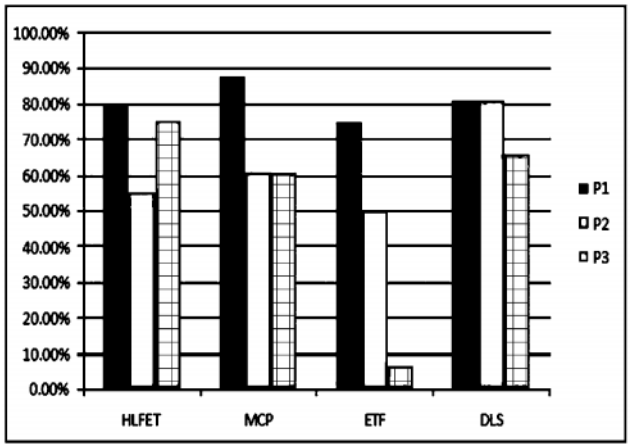

Fig. (ix): Processor Utilization for 15 Nodes

7. CONCLUSION

After Comparative analysis following results were derived:

- Makespan of ETF showed large increase in amount while increasing the tasks from 10 to 15 compared to other algorithms.
- The average processor utilization remained same for all algorithms with 5 tasks.MCP, ETF and DLS utilized processor efficiently than HLFET with 10 tasks. With 15 tasks, DLS proved to be better than other algorithms and ETF showed large drop in utilization rate.
- SLR remained almost the same with 5 and 10 tasks. With 15 tasks DLS was the one with lesser SLR.
- Same is the case with Speedup. With 5 and 10 tasks speedup of all the algorithms was same. With 15 tasks again DLS was the algorithm with higher SpeedUp.

So it can be concluded from above results that DLS is one of the efficient algorithms, considering the data gathered using the scenarios and the performance calculated from them.

7.1. Future Scope

A lot of work can be done considering more case scenarios:

- Heterogeneous environment can be considered in which multiple processors having different configuration are used.
- Combination of both Homogenous and Heterogeneous can be considered.
- The number of tasks can be changed to create test case scenarios.
- More algorithms can be considered and their performance with other can be estimated.

REFERENCES

[1] Ishfaq Ahmad and Min-You Wu, "Analysis, Evaluation and Comparison of Algorithm for Scheduling Task Graph on Parallel Processor" pp 1087-4087, IEEE 1996.

[2] Ishfaq Ahmad and Min-You Wu, "Performance Comparison of Algorithms for Static Scheduling of DAG to Multiprocessors", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8979&rep=rep1&type=pdf

[3] T.L. Adam, K. Chandy and J. Dickson, "A Comparison of List Scheduling for Parallel Processing Systems," Communications of the ACM, 17, No. 12, pp. 685-690, Dec. 1974.

[4] T.C. Hu, "Parallel Sequencing and Assembly Line Problems," Operations Research, 19, No. 6, pp. 841-848, Nov. 1961.

[5] Y.C. Chung and S. Ranka, "Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed-Memory Multiprocessors," Proc. of Supercomputing'92, pp. 5, 12-521, Nov. 1992.

[6] E.G. Coffman and R.L. Graham, "Optimal Scheduling for Two-Processor Systems," Acta Informatica, 1, pp. 200-213, 1972.

[7] G.C. Sih and E.A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," IEEE Trans, on Parallel and Distributed Systems, 4, No. 2, pp. 75-87, Feb. 1993.

[8] J. Baxter and J.H. Patel, "The LAST Algorithm: A Heuristic-Based Static Task Allocation Algorithm," Proc. of Int'l Conference on Parallel Processing, 11, pp. 217-222, Aug. 1989.

[9] Shiyuan Jin, Guy Schiavone, Damla Turgut, "A Performance Study of Multiprocessor Task Scheduling Algorithms" 43, pp. 77-97, Jan 2008.

[10] "Fast Scheduling and Partitioning Algorithm in the Multiprocessoe System with Redundant Communication Resources" www.springerlink.com/content/0np00wu1r0c7te8m/fulltext.pdf

[11] T. Hagras, J. Janecek, "Static vs. Dynamic List-Scheduling Performance Comparison" Acta Polytechnica, 43, No. 6/2003.

[12] "Scheduling Algorithms and Support Tools for Parallel Systems"www.fer.hr/_download/repository/Grudenickvalifikacijski.pdf.

[13] Kwok Y., Ahmed I.: Benchmarking the Task Graph Scheduling Algorithms. Proc. IPPS/SPDP, 1998

[14] "Approximation Algorithms for Multiprocessor Scheduling under Uncertainty" www.ccs.neu.edu/home/rraj/Pubs/uncertainty.pdf