ISSN 2063-5346

ESTIMATION OF DAG BASED SCHEDULING ALGORITHMS FOR BOUNDED NUMBER PROCESSORS

Suresh M. Wadaskar[1], Sunita Kushwaha[2]

Research Scholar, MATS School of Information Technology, MATS University, Raipur (C.G.), India

[1]mwsuresh99@gmail.com

[*]Associate Professor, MATS School of Information Technology, MATS University, Raipur (C.G.), India

[2]drsunitak@matsuniversity.ac.in

Article History: Received: 12.05.2023 **Revised:** 25.05.2023 **Accepted:** 05.06.2023

Abstract

Parallel computing is a promising approach to meet the computational requirement of large number of current and emerging applications. It is a technique of executing multiple tasks simultaneously on multiple processors. The main goal of parallel computing is to increase the computation speed. Efficient task scheduling and mapping are the big issues in homogeneous parallel computing environment. In this paper, several classes of algorithm are examine and then compare the performance of a class of scheduling algorithms known as the Bounded Number of Processors (BNP) scheduling algorithms. Therefore, four scheduling algorithms namely, HLFET, MCP, ETF and DLS are selected for evaluation. Evaluation is based on various scheduling parameters such as makespan, speedup etc. The focus of algorithms is to minimizing the total schedule length and increasing the efficiency of the system to improve the performance of system.

Keyword: BNP, Parallel computing, Scheduling, DAG, Homogeneous processors.

I. INTRODUCTION

Parallel computing is related to the application of many computers running in parallel to solve computationally intensive problems. One of the biggest issues in parallel computing is efficient task scheduling. Parallel computing is a technique of executing multiple tasks simultaneously on multiple processors. The main goal of parallel computing is to increase the speed of computation. Efficient task scheduling and mapping is the big issue in homogeneous parallel computing environment [1]. Directed Acyclic Graph (DAG) task models have been widely used in various areas to represent dependency such as in dependent task set. The efficiency of the system depends on the performance of processors [2][3]. Following are the number of problems and challenges poses in parallel processing.

a) Designing a parallel algorithm for the application,
b) Partitioning of the application into tasks,
c) Coordinating communication and synchronization,
d) Scheduling of the tasks/processes on to the processor (machine) [3].

Scheduling and allocation are highly important issue, since an inappropriate scheduling of tasks can fail to exploit the true potential of the system and can offset the gain from the parallelization, so the focus on the scheduling aspects is more important and motivates for the research in this field. The one of the objective of scheduling is to minimize the completion time of parallel application by properly allocating the tasks to the processors.

The scheduling problem exists in two forms: Static and Dynamic. The characteristics of a parallel program which are required to be known before scheduling or the program execution includes i) the task processing time, ii) its communication, iii) data dependencies and iv) synchronization so the parallel program can be represented by static model and the scheduling can be done at compile time (off-line)[1].

A parallel program or the static model can be represented by a node and edge-weighted Directed Acyclic Graph (DAG) in which the node-weight represents task processing time and the edge-weight denotes the data dependencies and communication times between the tasks [3]. DAG uses the two-tuple to represent the execution time and the communication time of the tasks. So the scheduling of tasks can be changed to the scheduling of DAG models [2][3].

The rest of this paper is organized as follows. In the next section, describe the generic DAG model and discuss its variations and techniques. A classification of scheduling algorithms is presented in Section III. The four BNP scheduling algorithms are discussed in Section IV. The performance result and comparisons are presented in Section V, Section VI concludes the paper.

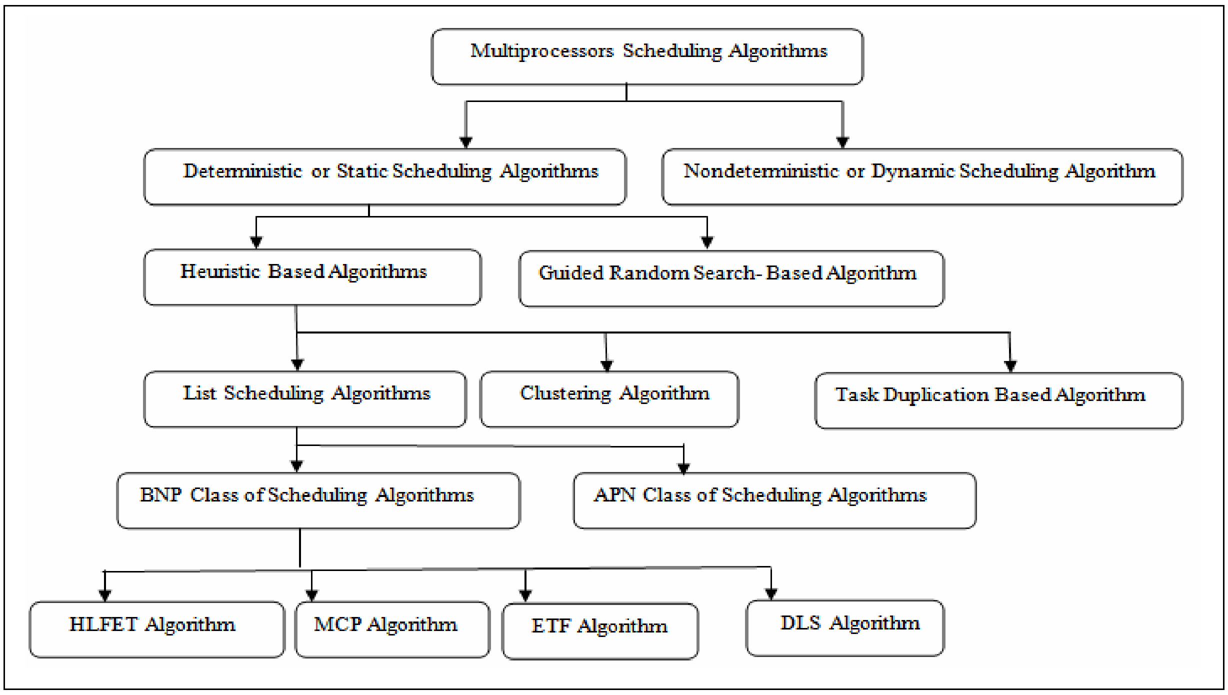
Multiprocessors Scheduling Algorithms
Deterministic or Static Scheduling Algorithms
Nondeterministic or Dynamic Scheduling Algorithm
Heuristic Based Algorithms
Guided Random Search-Based Algorithm
List Scheduling Algorithms
Clustering Algorithm
Task Duplication Based Algorithm
BNP Class of Scheduling Algorithms
APN Class of Scheduling Algorithms
HLFET Algorithm
MCP Algorithm
ETF Algorithm
DLS Algorithm

Fig1: Taxonomy of Multiprocessor Scheduling

II. DAG MODEL

DAG (Directed Acyclic Graph) represents parallel program ie. G(V, E), where V is a set of nodes and E is a set of directed edges. In the DAG, node represents task, which is set of instructions which must be executed sequentially without pre-emption in the same processor. Weight of a node n_i is called the computation/execution cost and is denoted by $W(n_i)$. The edges in the DAG, each of which is denoted by $E(n_i, n_j)$ corresponds to the communication messages and precedence constraints among the nodes. The weight of an edge is called the communication cost between two nodes and is denoted by $C(n_i, n_j)$. The source node of an edge is called parent node while the sink/destination node is called the child node. A node with no parents is called an entry node and a node with no child is called an exit node. The term node and tasks are same and are interchangeable [5][6].

A task cannot start its execution before it gathers all the information from its parent task. The communication cost between two tasks is assumed to be zero if both tasks are assigned to the same processor. If node n_i is scheduled on some processor then, $ST(n_i)$ and $FT(n_i)$ denotes the start-time and finish-time of n_i respectively. After all the nodes have been scheduled, the schedule length is defined as $\max_i\{FT(n_i)\}$ across all processors. The goal of scheduling is to minimize $\max i\{FT(n_i)\}$.The node and edge weights are usually obtained by estimation at the compile time[2][3][5].

III. BNP SCHEDULING ALGORITHMS

In this section, four well-known BNP scheduling algorithms namely HLFET, MCP, ETF and DLS are selected for evaluation. All these algorithms are for a limited number of homogeneous processors. The major characteristics of these algorithms are summarized [7].

A) The HLFET (Highest Level First with Estimated Times) Algorithm [3]: It is one of the simplest scheduling algorithms. The algorithm is briefly described below.

1) Calculate the static b-level of each node.

2) Make a ready list in a descending order of static b-level. Initially, the ready list *contains only the entry nodes. Ties are broken randomly.*

Repeat

3) Schedule the first node in the ready list to a processor that allows the earliest execution, using the non-insertion approach.

4) Update the ready list by inserting the nodes that are now ready.

Until all nodes are scheduled.

B) MCP (Modified Critical Path) Algorithm [3]: The MCP algorithm uses the ALAP of a node as the scheduling priority. The MCP algorithm first computes the ALAPs of all the nodes, then construct a list of nodes in an ascending order of ALAP times. Ties are broken by considering the ALAP times of the children of a node. The MCP algorithm then schedules the node on the list one by one such that a node is schedule to a processor that allows the earliest start time using the insertion approach. The algorithm is briefly described below.

1) Compute the ALAP time of each node.

2) For each node, create a list which consists of the ALAP times of the node itself and all its children in a descending order.

3) Sort these lists in an ascending lexicographical order. Create a node list according to this order.

Repeat

4) Schedule the first node in the node list to a processor that allows the earliest execution, using the insertion approach.

5) Remove the node from the node list.

Until the node list is empty.

C) The ETF (Earliest Time First) Algorithm [3]: The ETF algorithm computes, at each step, the earliest start times for all ready nodes and then selects the one with the smallest start-time. When two nodes have the same value in their earliest start-times, the ETF algorithm break the tie by scheduling the one with higher static priority. The ETF algorithm is briefly described below.

Step-1) Compute the static b-level of each node.

Step-2) Initially, the pool of ready nodes includes only the entry nodes.

Repeat

Step-3) Calculate the earliest start-time on each processor for each node in the ready pool. Pick the node-processor pair that gives the earliest time using the non-insertion approach. Ties are broken by selecting the node with a higher static b-level. Schedule the node to the corresponding processor.

Step-4) Add the newly ready nodes to the ready node pool.

Until all nodes are scheduled.

4) The DLS (Dynamic Level Scheduling) Algorithm [3]: This algorithm uses an attribute called dynamic level (DL), which is the difference between the static b-level of a node and its earliest start-time on a processor. At each scheduling step, the algorithm computes the DL for every node in the ready pool on all processors. The node-processor pair which gives the largest value of DL is selected for scheduling. The algorithm is briefly described below.

Step-1) Calculate the static b-level of each node.

Step-2) Initially, the ready list contains only the entry nodes.

Repeat

Step-3) Calculate the earliest start-time for every ready node on each processor. Hence, compute the DL of every nod-processor pair by subtracting the earliest start-time from the nodes static b-level.

Step-4) Select the node-processor pair that gives the largest DL. Schedule the node to the corresponding processor.

Step-5) Add the newly ready nodes to the ready pool.

Until all nodes are scheduled

IV. RELATED WORK

Directed Acyclic Graph (DAG) task models have been widely used in multi-core processor based parallel system. The efficiency of the system depends on the performance of multi-core processors. The proposed scheduling algorithm is the combines the cluster-based and the interval insertion strategies, to reduce the schedule length and number of processors. Proposed algorithm improves the efficiency of multi-core processors where scheduling of DAG tasks is required [8].

Heuristic task based on Critical Path and Task Duplication scheduling algorithm (HCPTD) is a combination of the table scheduling and the task replication scheduling. The algorithm improves the calculation method of task weight, and gives the scheduling sequence according to either it is mission-critical or descending order of weight. In this algorithm, processor chooses the earliest task completion time and the shortest task to exit-node distance. HCPTD algorithm can effectively improve the performance of scheduling in distributed systems [9].

Heterogeneous Earliest-Finish-Time (HEFT) and the Critical Path on a Processor (CPOP) algorithm are the two scheduling algorithms for a bounded number of heterogeneous processors with an objective to simultaneously meet high performance and fast scheduling time. Based on experimental study, the HEFT algorithm significantly outperformed the other algorithms in terms of cost metrics and performance, including frequency of best results, speedup, average schedule length ratio, and average running time [3].

Recursive task scheduling algorithm for a bounded number of heterogeneous processors runs on the network of Heterogeneous systems. Task-prioritizing phase, processor selection phase and moving phase are the three phases of tasks scheduling algorithm that (1) task prioritising phase assigns priority to tasks, (2) processor selection phase schedules tasks onto the processors which gives the latest start time and (3) moving phase is to move all the possible tasks until the starting time of the entry task is zero. The normal list scheduling algorithm selects tasks from the entry task to exit task while scheduling it onto the processors but the recursive algorithm selects the tasks from the exit task to entry task to schedules it onto the processors. In terms of performance metrics (i.e. schedule length ratio, speedup, efficiency and frequency of best results) and a cost metric (i.e. scheduling time) the recursive algorithm gives high performance. The performance has been compared with Iterative List Scheduling (ILS) algorithm and Heterogeneous Earliest Finish Time (HEFT) [10].

The proposed Heterogeneous Scheduling algorithm with Improved task Priority (HSIP) evaluated using some DAG based real application and compared it with scheduling algorithms such as, Predict Earliest Finish Time (PEFT), SD-Based Algorithm for Task Scheduling (SDBATS), Heterogeneous Earliest Finish Time (HEFT) and Critical Path on a Processor (CPOP). It has been observed that HSIP perform better in terms of scheduling time and performance metrics (that are average schedule length ratio, speedup and efficiency). The simulation result also confirms that HSIP algorithm is substantially better than the existing algorithms such as PEFT, SDBATS, CPOP and HEFT [2].

V. METHODOLOGY AND EXPERIMENTATION

Communication time and task precedence are the most important factor for scheduling in multiprocessor parallel system. Optimization of communication time between two tasks can result better performance in terms of the scheduling length, the efficiency and processor utilization. Communication to computation ratio is also one of the important factors to improve performance of the algorithm.

The very common factor/fact in every BNP class list scheduling algorithm is that after preparation of the priority list, schedule the first task (ie. Entry task) to the available processor which allows earliest start time OR earliest execution time. The earliest start time and earliest execution time both works in same way in BNP class for homogeneous processor but it differs in heterogeneous processor environment.

This work is focusing on the BNP class for homogeneous processor therefore; the entry node allocation is very important process and same for all algorithms in BNP. After allocation of Entry Task to the processor and when the entry (first) node starts its execution on the processor selected during processor selection, all the remaining processors are compulsorily idle till entry task complete its execution. After finishing execution of entry task or the first task on any of the selected processor, all the children of entry task will be waiting in the queue for scheduling. Parent task execution is completed on the one selected processor (eg. P1) and now more than one children (successors) are ready for processing.

Now, all the children (successor) of entry task requires communication time if they are scheduled on any of the available processor except the processor which was used by entry task (parent task) for schedule. So, if entry task has more than one children, then only one successor task can save its communication time by scheduling on same processor where entry task (parent task) was executed, and all other tasks required communication time for execution if they assigned processor other than the processor used by entry task.

VI. RESULTS ANALYSIS

In this section, the performance results and comparisons of selected BNP scheduling algorithms discussed. Performance measurement in the term of makespan, speedup and NLS are calculated. Some DAG based BNP scheduling namely, HLFET, MCP, ETF and DLS are coded in MATLAB and simulation has been performed [9][11-15].

A. Performance Parameters

1. Makespan: Makespan is defined as the completion time of the algorithm. It is calculated by measuring the finishing time of the exit task by the algorithm.

2. Speed Up: The Speed Up value is computed by dividing the sequential execution time by the parallel execution time [14][15].

3. Normalized schedule length (NSL): Schedule length of scheduling algorithm divided by the sum of weights of the Task on the original critical-path.

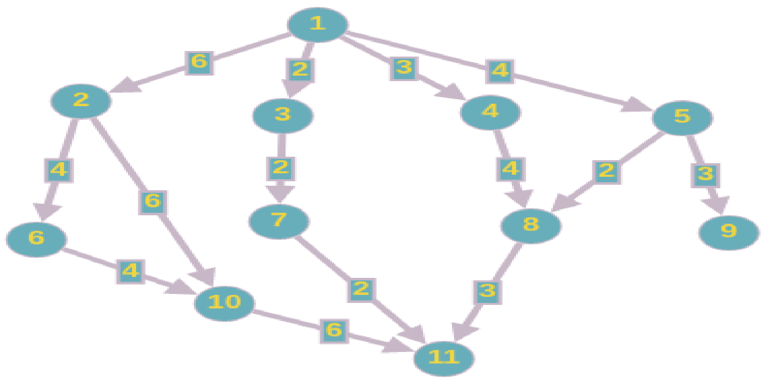

Fig2: DAG Model for **11 tasks**

Table1: TASK MATRIX OF 11 TASK DAG MODEL

Computation Time	Tasks	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	
2	T1	0	6	2	3	4	0	0	0	0	0	0	Critical Path = 34
4	T2	0	0	0	0	0	4	0	0	0	6	0	
4	T3	0	0	0	0	0	0	2	0	0	0	0	
5	T4	0	0	0	0	0	0	0	4	0	0	0	
4	T5	0	0	0	0	0	0	0	2	3	0	0	
3	T6	0	0	0	0	0	0	0	0	0	4	0	CCR= 0.095865
2	T7	0	0	0	0	0	0	0	0	0	0	2	
5	T8	0	0	0	0	0	0	0	0	0	0	3	
4	T9	0	0	0	0	0	0	0	0	0	0	0	
3	T10	0	0	0	0	0	0	0	0	0	0	6	
2	T11	0	0	0	0	0	0	0	0	0	0	0	

B. Experimental Setup

The series of experiment has been done for performance evaluation therefore, performance parameters are calculated for above mentioned DAG based BNP scheduling algorithm. Experimental setup are set for simulation is as follows :- processing time is between from 1 time unit to 10 time unit and arrival time is also 1 to 10 time unit, number of processors are 4 and number of dependent tasks/ processes is 11, Entry task and exit task is single task.

Table2: Schedule created by HLFET

Step	Start Time	Task No	Execution Time	Finish Time	Processor No
1	0	1	2	2	1
2	2	2	4	6	1
3	5	4	5	10	2
4	6	5	4	10	1
5	4	3	4	8	3
6	10	6	3	13	1
7	12	8	5	17	2
8	13	10	3	16	1
9	8	7	2	10	3
10	13	9	4	17	3
11	20	11	2	22	1

algorithm for Fig2 Task Graph

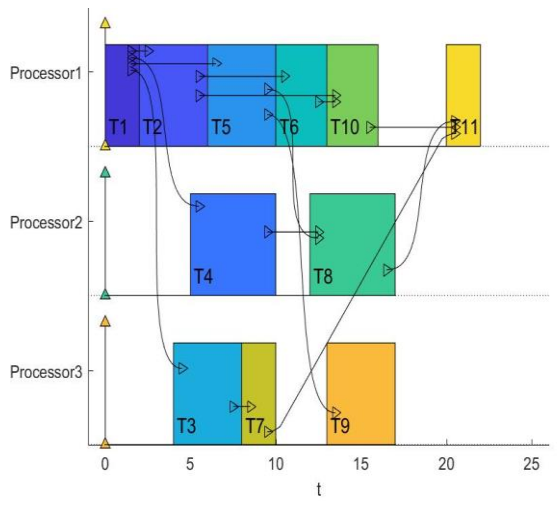

Fig3: Schedule created by HLFET algorithm for Fig2 Gant Chart

Table3:Schedule created by MCP algorithm for Fig2 Task Graph

Step	Start Time	Task No	Execution Time	Finish Time	Processor No
1	0	1	2	2	1
2	2	2	4	6	1
3	5	4	5	10	2
4	6	6	3	9	1
5	6	5	4	10	3
6	4	3	4	8	4
7	9	10	3	12	1
8	12	8	5	17	2
9	8	7	2	10	4
10	10	9	4	14	3
11	18	11	2	20	2

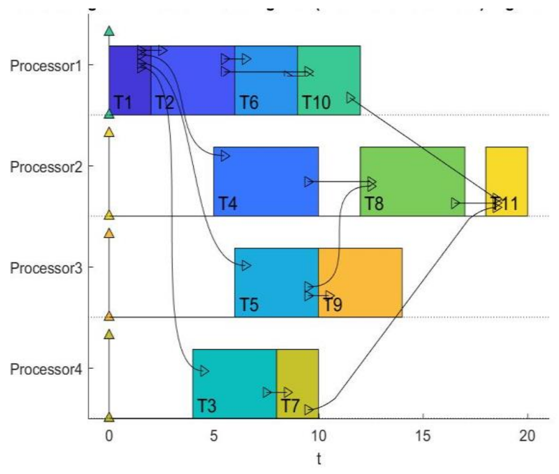

Fig4: Schedule created by HLFET algorithm for Fig2 Gant Chart

Table4: Schedule created by ETF algorithm for Fig2 Task Graph

Step	Start Time	Task No	Execution Time	Finish Time	Processor No
1	0	1	2	2	1
2	2	3	4	6	1
3	5	4	5	10	2
4	6	5	4	10	1
5	8	2	4	12	3
6	8	7	2	10	4
7	10	9	4	14	1
8	12	8	5	17	2
9	12	6	3	15	3
10	15	10	3	18	3
11	20	11	2	22	3

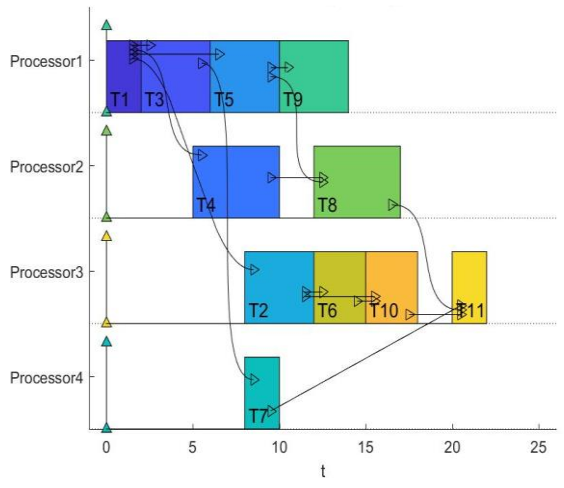

Fig5: Schedule created by ETF algorithm for Fig2 Gant Chart

Table5: Schedule created by DLS algorithm

Step	Start Time	Task No	Execution Time	Finish Time	Processor No
1	0	1	2	2	1
2	2	4	5	7	1
3	6	5	4	10	2
4	7	2	4	11	1
5	4	3	4	8	3
6	8	7	2	10	3
7	11	8	5	16	2
8	11	6	3	14	1
9	13	9	4	17	3
10	14	10	3	17	1
11	19	11	2	21	1

for Fig2 Task Graph

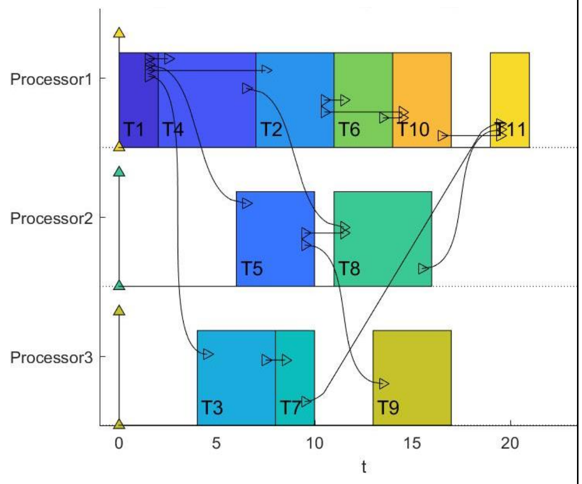

Fig6: Schedule created by DLS algorithm for Fig2 Gant Chart

Table6: Priority Attributes of 11 Task DAG Model (Figure-2)

Task No.	Burst Time	Top Level	Bottom Level	Static Level	ALAP Time	Dynamic Level
1	2	0	34	14	0	14
2	4	8	26	12	8	4
3	4	4	12	8	22	4
4	5	5	19	12	15	7
5	4	6	16	11	18	5
6	3	16	18	8	16	-8
7	2	10	6	4	28	-6
8	5	14	10	7	24	-7
9	4	13	4	4	30	-9
10	3	23	11	5	23	-18
11	2	32	2	2	32	-30

Table7: performance metrics for all five algorithm

Algorithm	MakeSpan	Speed Up	NSL
HLFET	22.00	1.73	0.65
MCP	20.00	1.90	0.59
ETF	22.00	1.73	0.65
DLS	21.00	1.81	0.62

Table 2 present the scheduling sequence of the dependent task set according to HLEFT on particular a processor and fig3 shows the Gantt chart for the same. Table 3 present the scheduling sequence of the dependent task set according to MCA on particular a processor and fig4 shows the Gantt chart for the same. Table 4 present the scheduling sequence of the dependent task set according to ETF on particular a processor and fig5 shows the Gantt chart for the ETF. Table 5 shows the scheduling sequence of the dependent task set according to DLS on particular a processor and fig6 shows the Gantt chart for the same.

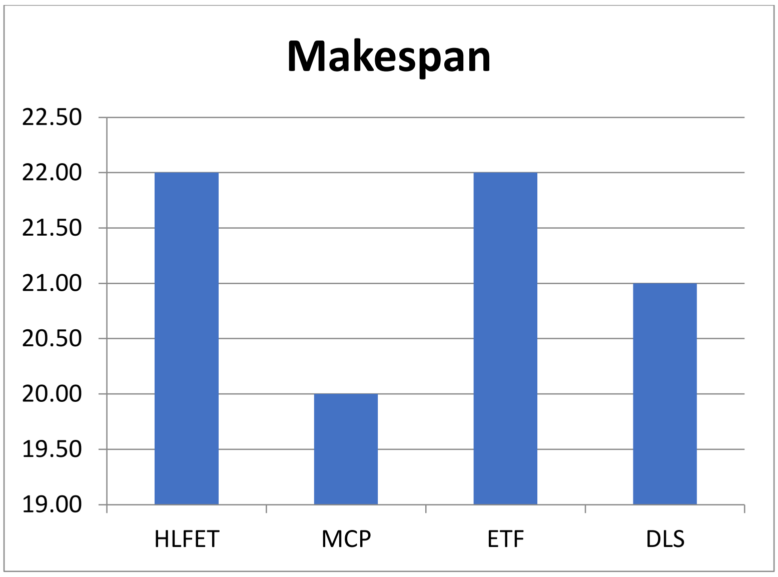

Fig7: MAKESPAN for Scheduling Algorithm

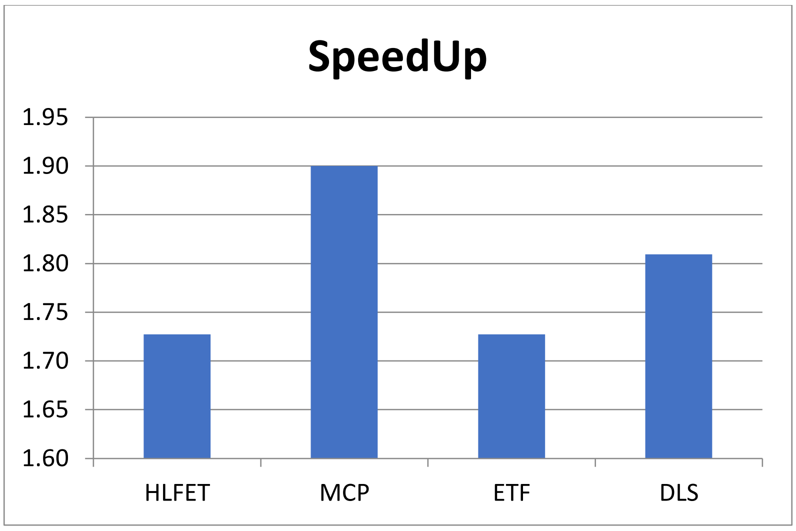

Fig8: SpeedUp for Scheduling Algorithm

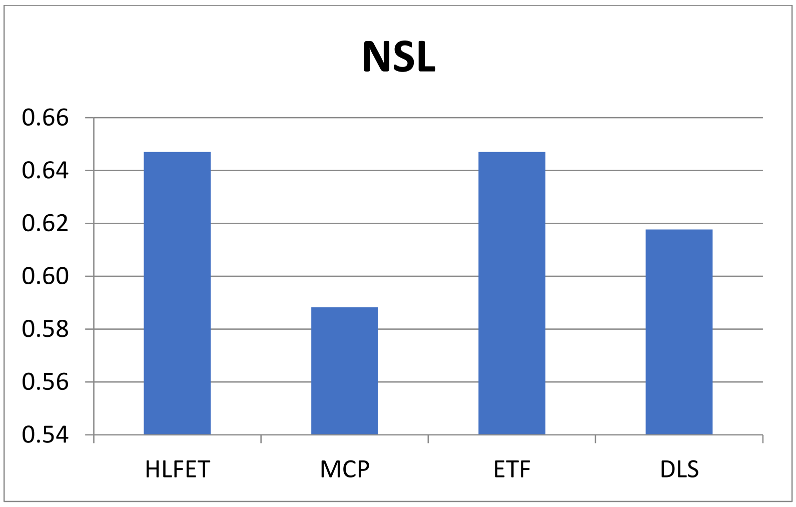

Fig9: NLS for Scheduling Algorithm

Performance of HLEFT, MCP, EFT and DLS are shown with the help of figures: fig7, fig8 , fig9 and Table7 all the parameter calculated with the help of MATLAB script. According to our simulation MCP perform best as compare to other as the Makespan and NSL of MCP is Minimum but speedup is higher than other selected BNP scheduling algorithms.

Observation: The entry node allocation is very important process and same for all algorithms in BNP. After allocation of Entry Task to the processor and when the entry (first) node starts its execution on the processor selected during processor selection, all the remaining processors are compulsorily idle till entry task complete its execution. After finishing execution of entry task or the first task on any of the selected processor, all the children of entry task will be waiting in the queue for scheduling. Parent task execution is completed on the one selected processor (eg. P1) and now more than one children (successors) are ready for processing. Now, all the successor of entry task requires communication time if they are scheduled on any other available processor except the processor which was used by entry task (parent task) for schedule. Therefore, if entry task has more than one child, then only one successor task can save its communication time by scheduling on same processor where entry task was executed, and all other tasks required communication time for execution if they assigned processor other than the processor used by entry task.

VII. CONCLUSION

In this era parallel and distributed computing highly recommended in various application for computational work. Parallel and distributed computing completes more computation in less time by proper work distribution among processors. This work distribution carried out with the help of scheduling which is one of the challenging areas of computing system. There are various classes of scheduling exist according to system and task characteristics, one of them are BNP class of scheduling where bounded numbers of processor work together to complete the assigned tasks which are mostly required proper communication for their execution. In the BNP class of scheduling HLFET, DLS, MCP and ETF are widely used in diverse scenario for study and research purpose. Most of the research work focused on homogeneous systems and dependent task model. This research work perform the simulation of above mentioned DAG based scheduling algorithm for BNP, for the performance parameters Makespan, speedup and NLS. Performance of MCP is better than other selected DAG based scheduling for BNP as it utilizes the time better than others.

REFERENCES

[1] Singh, E. N., Kaur, G., Kaur, P., & Singh, G. (2012). “Analytical performance comparison of BNP scheduling algorithms” Global Journal of Computer Science and Technology, Volume 12 Issue 10, PP 1-9.

[2] Liu Yuan, Pingui Jia and Yiping Yang, “Efficient scheduling of DAG tasks on multi-core processor based parallel systems”, TENCON 2015 - 2015 IEEE Region 10 Conference, Macao, 2015, pp. 1-6.

[3] Yu-Kwong Kwok, Ishfaq Ahmad “Static Scheduling Algorithm for Allocating Directed Task Graph to multiprocessors”, ACM Computing Surveys, Vol. 31,no. 4, December 1999.

[4]Gurjit Kaur, A DAG based Task Scheduling Algorithms for Multiprocessor System - A Survey, International Journal of Grid and Distributed Computing Vol. 9, No. 9 (2016), pp.103-114.

[5] Al Ebrahim, S., Ahmad, I., “Task scheduling for heterogeneous

computing systems", J Supercomput., vol 73, pp. 2313-2338 (2017).

[6] Popa, E., Iacono, M., Pop, F., "Adapting MCP and HLFET Algorithms to Multiple Simultaneous Scheduling", Int J Parallel Prog vol 46, pp. 607–629 (2018).

[7] Parneet Kaur, Dheerendra Singh, Gurvinder Singh, "Analysis, Comparison and Performance Evaluation of BNP Scheduling Algorithms in Parallel Processing", International Journal of Information Technology and Knowledge Management January-June 2011, Volume 4, No. 1, pp. 279-284.

[8] Ranjit Rajak, COMPARISON OF BOUNDED NUMBER OF PROCESSORS (BNP) CLASS OF SCHEDULING ALGORITHMS BASED ON MATRICES, GESJ: Computer Science and Telecommunications 2012, volume .2, issue 34.

[9] Arora, Nidhi, Navneet Singh and Parneet Kaur, "Performance Comparison of BNP Scheduling Algorithms In Homogeneous Environment.", Global journal of computer science and technology, vol 12 (2012): pp 1-9

[10] Sharma, A., Kaur, H. (2013). QOS, Comparison of BNP Scheduling Algorithms with Expanded Fuzzy System. International Journal of Advanced Computer Science and Applications, 4(5).

[11] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, March 2002, pp. 260-274.

[12] Garg, Akanksha & S. Sethi, Navdeep Arora, Nidhi, Makkar, Amit. (2013), "BNP Task Scheduling Algorithms for Performance Evaluation In Parallel System", International Journal of Computers Technology, vol 12, pp 3768-3777.

[13] I. Ahmad, Yu-Kwong Kwok and Min-You Wu, "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors", Proceedings Second International Symposium on Parallel Architectures, Algorithms, and Networks, Beijing, China, 1996, pp. 207-213.

[14] Sunita Kushwaha, Varsha Thakur, "Heuristic Oriented Process Scheduling for Homogeneous Multiprocessor Environment, International Journal of Mechanical Engineering, Vol.7 No.5 (May, 2022), pp715-726.

[15]Sunita Kushwaha, Sanjay Kumar, "Performance Analysis of List Scheduling on Homogeneous Multiprocessors system", International Journal of Computer Applications, Volume 152 – No.7, October 2016, pp 29-31.