Available online at www.pelagiaresearchlibrary.com

Pelagia Research Library

Advances in Applied Science Research, 2014, 5(3):280-285

ISSN: 0976-8610
CODEN (USA): AASRFC

Comparative analysis of heuristics for multiprocessor task scheduling problem with homogeneous processors

Sunita Dhingra[1]*, Satinder Bal Gupta[2] and Ranjit Biswas[3]

[1]*Department of Computer science & Engineering, University Institute of Engineering & Technology, Maharshi Dayanand University Rohtak, Haryana, India*
[2]*Department of Computer Science, Vaish College of Engineering, Rohtak, Haryana, India*
[3]*Department of Computer Science & Engineering, Jamia Hamdard University, New Delhi, India*

ABSTRACT

Scheduling of a task on a multiprocessor system represented by a directed acyclic graph for minimizing the different performance measures is a prominent problem in parallel processing. As judgment of an optimal schedule for multiprocessor task scheduling problem is a NP hard problem and different researchers have resorted for devising efficient heuristics. List scheduling heuristics belong to one of the categories used for multiprocessor task scheduling problem. Present work considers the comparative analysis of five commonly used list scheduling heuristics based on makespan and total completion time of the schedule for homogeneous multiprocessors. A defined Performance Index (PI) is used for the comparative analysis of different heuristics and it has been proved that the Insertion Scheduling Heuristic (ISH) Algorithm and Earliest Time First (ETF) Algorithm provides the best results for trade-off between the makespan and total completion time of the schedule.

Keywords: Multiprocessor task scheduling, Makespan, Total completion time, Heuristics

---

INTRODUCTION

Proficient scheduling of computationally intensive programs is the most important and complicated matter of subject for achieving higher performance in a parallel computing. A program can be decomposed into a set of smaller tasks having dependency and precedence requirements. The aim is to assign tasks among available processors in such a way that the precedence requirements between tasks can be satisfied along with optimisation of different performance measures. The performance measures can be the minimisation of the overall length of time required for executing the entire program i.e. the schedule length (makespan), the total completion time and so on.

Finding an optimal solution for the multiprocessor task scheduling problem is a NP-hard [1] and numbers of heuristics, randomized and exact methods have been developed by several researchers for solving these NP hard problems. It has been established that finding optimality to NP hard problems is not a viable option as large amount of computational time is required for judgment of such solutions. In reality, a good initial solution can be obtained by a heuristic in a reasonable computational time. Heuristics used for multiprocessor task scheduling problem are normally divided into three categories i.e. list scheduling, clustering based heuristics and duplication based heuristics.

List scheduling techniques are generally used for task scheduling problems that allot a priority to the tasks. As a processor becomes available, the highest priority task in the task list is allocated to the processor and removed from the list. Selection of candidate tasks can be random or based on some rule if more than one task has the same priority. Generally, characteristics those are employed for assigning priority are the b-level (bottom level), t-level (top level), static level (sl) and ALAP (As-Late-As-Possible) start-time.

Adam et al. [2] proposed Highest Level First with Estimated Times (HLFET) Algorithm which is the simplest list scheduling algorithm that uses static b-level as node priority. It assigns the task to the processor according to minimum start time. HLFET uses no-insertion approach i.e. an idle time slot is not utilized, which affects the performance. The Insertion Scheduling Heuristic (ISH) algorithm, proposed by Kruatrachue and Lewis [3], improves the HLFET algorithm by utilizing the idle time slots in the scheduling. Initially, it uses the same approach as HLFET to make a ready list based on static *b-level* and schedule the first node in the ready list using the non-insertion approach. The difference is that, once the scheduling of this node creates an idle slot, ISH checks if any task in the ready list can be inserted into the idle slot but cannot be scheduled earlier on the other processors. The algorithm schedules such tasks as many as possible into the idle slot [4].

Hwang et al. [5] proposed the ETF (Earliest Time First) algorithm and computes the earliest start-times at each step for all ready nodes and chooses the one with the smallest start-time. The earliest start time of a node can be computed by examining the start-time of the node on all processors exhaustively. When two nodes have the same value in the earliest start-times, the ETF algorithm breaks the tie by scheduling the one with the higher static priority. Wu and Gajski [6] developed Modified Critical Path (MCP) algorithm that uses the ALAP of a node as the scheduling priority. The MCP algorithm first computes the ALAPs of all the nodes and then constructs 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 schedules the nodes on the list (one by one) such that a node is scheduled to a processor that allows the earliest start time using the insertion approach. Sih and Lee [7] proposed the DLS algorithm that used 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 the processors. The node-processor pair which provides the largest value of DL is selected for the scheduling.

Kwok and Ahmad [8] presented a comprehensive review and classification of deterministic static scheduling algorithms and compared different scheduling algorithms for a nine-task problem. Davidovic et al. [9] focused on the comparison of list scheduling approaches and proposed a single pass deterministic algorithm, chaining, based on list scheduling techniques.

In the present work, comparative analysis of the commonly used list scheduling heuristics has been done for makespan and total completion time criteria. The steps of different heuristics considered in the present work are described in table 1.

The standard multiprocessor task scheduling problems with communication cost have been considered for the comparative analysis. Finally, the analysis of scheduling heuristics for the makespan and total completion time criteria with variation in number of processors has been done. The next sections consider the materials and methods, results and discussion followed by conclusion.

Table 1: Steps of HLFET, ISH, ETF, MCP and DLS heuristics

Highest Level First with Estimated Times (HLFET)	Insertion Scheduling Heuristic (ISH)	Earliest Time First (ETF)	Modified Critical Path (MCP)	Dynamic Level Scheduling(DLS)
1. Calculate the static b-level (i.e., SL or static 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 start time, using the non insertion approach. 4. Update the ready list by inserting the nodes that are now ready u**ntil** all nodes are scheduled.	**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 the processor that allows the earliest start time, using the non insertion algorithm. **4.** If scheduling of this node causes an idle time slot, then find as many nodes as possible from the ready list that can be scheduled to the idle time slot but cannot be scheduled earlier on other processor. **5.** Update the ready list by inserting the nodes that are now ready u**ntil** all nodes are scheduled.	**1.** Compute the static b-level of each node. **2.** Initially, the pool of ready nodes includes only the entry nodes. **Repeat** **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. **4.** Add the newly ready nodes to the ready node pool u**ntil** all nodes are scheduled.	**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 u**ntil** the node list is empty.	1. Calculate the b-level of each node. **2.** Initially, the ready node pool includes only the entry nodes. **Repeat** **3.** Calculate the earliest start-time for every ready node on each processor. Compute the DL of every node-processor pair by subtracting the earliest start-time from the node's static b-level. **4.** Select the node-processor pair that gives the largest DL. Schedule the node to the corresponding processor. **5.** Add the newly ready nodes to the ready pool **until** all nodes are scheduled.

MATERIALS AND METHODS

Multiprocessor task scheduling problem has been considered in which some task are dependent on other tasks & cannot be started until the predecessors have been processed. After a task is processed, its successor task may be processed only after a predefined time called as communication cost [10]. Input is considered in terms of Directed acyclic Graph (DAG) for this dependency. In a DAG, G = (V, E), V the set of vertices represent the tasks & E is the set of directed edges for demonstrating the dependency between tasks. The computation weight of each vertex show the number of CPU cycles required by a task & the computation weight on each directed edge shows the communication cost.

The present work is based on the deterministic model, i.e. the number of processors, the execution time of tasks, the relationship among tasks and precedence constraints etc. are known in advance. In addition, the communication cost between two tasks has been considered and the tasks are non-preemptive, i.e. the current task completes before the execution of new task. The multiprocessor system consists of a set of homogeneous processors. Heuristic comparison is based on makespan and total completion time.

The Makespan of a schedule is the time at which the last task completes for a particular schedule i.e. C_{max}. Total completion time of a schedule is calculated as $\sum_{i=1}^{n} Ci$ where C_i is the completion time of i_{th} task of a schedule. Minimising both the makespan and total completion time is required for effective utilisation of processors and proper load balancing. Load balancing is used for optimizing the resource use, maximize throughput, minimize response time and avoid overload of any one of the resources.

RESULTS AND DISCUSSION

In this study, five list scheduling heuristics i.e. HLFET, ISH, MCP, ETF and DLS have been compared for different multiprocessor task scheduling problems on homogeneous processors. The algorithm of the heuristics has been coded in MATLAB environment. Makespan and total completion time are the two performance measures which have been considered for the comparative analysis. The evaluation of all the heuristics is carried out using the standard problems of multiprocessor task scheduling as shown in table 2.

Table 2: Task Scheduling Problems used for Comparative Analysis

Problems	No. of Tasks	No. of processors considered	Communication cost	Reference	Remarks
T9	9	2,3,4	Variable	Bonyadi and Moghaddam [10]	
T14_1	14	2,3,4	Fixed(20)	Tsuchiya et al.[11]	LU decomposition
T14_2	14	2,3,4	Fixed(80)	Tsuchiya et al.[11]	LU decomposition
T16_1	16	2,3,4	Fixed(40)	Wu and Gajski[6]	Laplace equation solver
T16_2	16	2,3,4	Fixed(160)	Wu and Gajski[6]	Laplace equation solver
T18	18	2,3,4	Variable	Bonyadi and Moghaddam [10]	Gaussian Elimination

The comparative analysis has been done by defined Performance Index (PI) which can be computed as:-

$$\text{Performance Index } (PI) = 1 - \frac{Heuristic_{sol} - Best_{sol}}{Best_{Sol}}$$

Where $Heuristic_{sol}$ is the solution obtained by a given heuristic and $Best_{sol}$ is the best solution obtained among all the heuristics for a particular problem considered irrespective to number of processors. PI nearer to unity provides the best results.

From table 3 and 4, it has been found that, the solution quality of heuristics is dependent on number of tasks and processors. It has been known the makespan and total completion time of the schedule should be minimum for achieving maximum efficiency, effective utilisation of processors, maximum throughput and proper load balancing. There should be trade-off between the solution provided by the heuristics for makespan and total completion time. After comparing the different heuristics, the ISH heuristic provides the best average PI for makespan of 0.943 and ETF heuristic for total completion time with PI of 0.974. Therefore for trade-off between the two performance measures, ISH and ETF heuristics provides the best results, especially for larger and complex multiprocessor scheduling problems.

Table 3: Performance Index (PI) for Makespan

Problem	Processors	HLFET	MCP	DLS	ISH	ETF
T9	2	**1**	0.79	**1**	0.92	**1**
	3	**0.96**	0.79	0.79	**0.96**	0.79
	4	**0.96**	0.79	0.79	**0.96**	0.79
T14_1	2	**0.96**	0.90	**0.96**	**0.96**	**0.96**
	3	**1**	**1**	**1**	**1**	**1**
	4	**1**	**1**	**1**	**1**	**1**
T14_2	2	**1**	0.93	0.93	**1**	**1**
	3	**0.98**	0.93	0.93	0.93	0.91
	4	**0.98**	0.93	0.93	0.93	0.91
T16_1	2	**0.89**	**0.89**	**0.89**	**0.89**	**0.89**
	3	**1**	**1**	**1**	**1**	0.96
	4	**1**	**1**	**1**	**1**	0.96
T16_2	2	0.92	**1**	**1**	**1**	**1**
	3	0.88	**1**	**1**	**1**	**1**
	4	0.88	**1**	**1**	**1**	**1**
T18	2	0.81	0.89	0.81	0.81	**1**
	3	0.81	0.81	0.81	0.81	**0.89**
	4	0.81	0.81	0.81	0.81	**0.89**
Average		0.936	0.914	0.925	0.943	0.942
Ranking		**3**	**5**	**4**	**1**	**2**

Table 4: Performance Index (PI) for Total Completion Time

Problem	Processors	HLFET	MCP	DLS	ISH	ETF
T9	2	**0.97**	0.93	0.96	**0.97**	0.96
	3	**1**	0.96	0.96	**1**	0.96
	4	**1**	0.96	0.96	**1**	0.96
T14_1	2	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**
	3	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**
	4	**1**	**1**	**1**	**1**	**1**
T14_2	2	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
	3	0.95	**1**	**1**	**1**	**1**
	4	**1**	**1**	**1**	**1**	**1**
T16_1	2	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**
	3	**1**	**1**	**1**	**1**	0.98
	4	**1**	**1**	**1**	**1**	0.98
T16_2	2	0.93	**1**	**1**	**1**	**1**
	3	0.94	**1**	**1**	**1**	**1**
	4	0.94	**1**	**1**	**1**	**1**
T18	2	0.82	0.85	0.87	0.85	**1**
	3	0.86	0.87	0.90	0.88	**0.96**
	4	0.89	0.91	0.90	0.91	**0.98**
Average		0.947	0.957	0.961	0.964	0.974
Ranking		**5**	**4**	**3**	**2**	**1**

Figure 1 and 2 shows the analysis of Performance Index (PI) with 2, 3 and 4 processors for different considered multiprocessor task scheduling problems for makespan and total completion time respectively. It can be seen that the ETF for 2 processors shows superiority over others for all the considered problems with makespan and total completion time respectively. The solution quality of the heuristics depend on number of task for 3 and 4 processors, as ISH for 9 and 14 tasks, HLFET and ISH for 16 task problems and ETF for increased task size problems provides the compromise results.

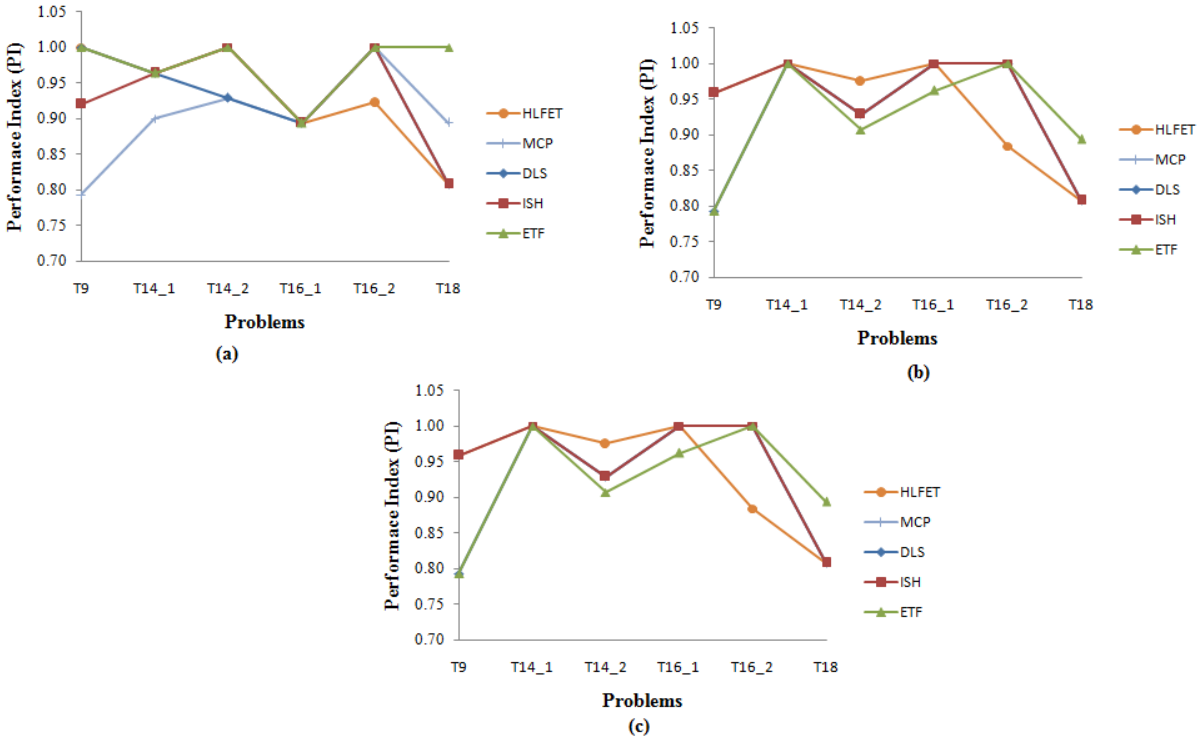

Figure 1: Comparison of different heuristics for task scheduling problems for Makespan
(a) Two Processor (b) Three Processor (c) Four Processor

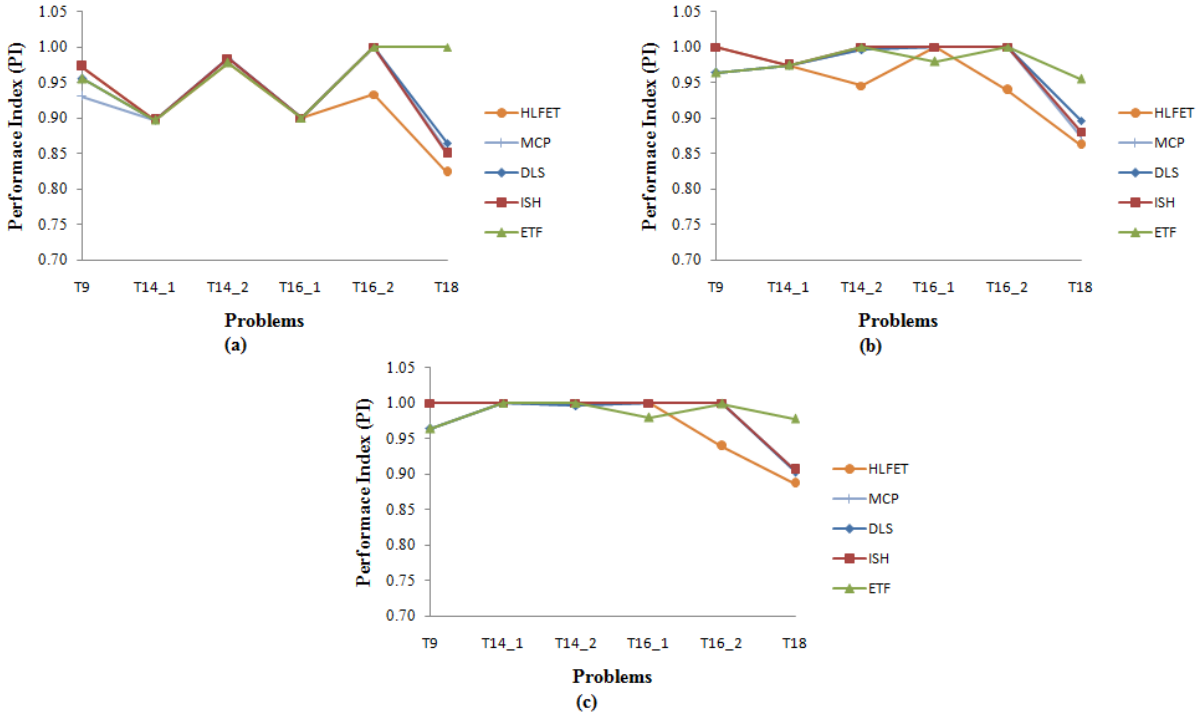

Figure 2: Comparison of different heuristics for task scheduling problems for Total Completion Time (a) Two Processor (b) Three Processor (c) Four Processor

Therefore, ISH and ETF heuristics provide the best trade-off results and can be effectively used for minimised makespan and total completion time for the multiprocessor task scheduling problems.

CONCLUSION

Present work considers the comparison of five commonly used list scheduling heuristics (HLFET, MCP, DLS, ISH and ETF) for multiprocessor tasks for the makespan and total completion time performance measures. Makespan

and total completion time are the performance measures that show the effective system utilisation and proper load balancing. There should be trade-off between the heuristics for providing the best solution in terms of two performance measures. The comparative analysis has been made by the defined Performance Index (PI) on standard problems upto 18 tasks and 4 processors with communication cost on homogeneous processors and shows that the ISH and ETF provides the best results as compared to others. Also for minimum number of processors required, ETF provides the best trade-off results for most of the task scheduling problems.

REFERENCES

[1] Hou E S, Ansari N, Ren H, *IEEE Transactions on Parallel and Distributed Systems*, **1994**, 5(2), 113 – 120.
[2] Adam T L, Candy K M, Dickson J, *Communication ACM*, **1974**, 17(12), 685-690.
[3] Kruatrachue B, Lewis T G, *Technical Report*, **1987**, 87-60-3, Oregon State Univ.
[4] Jin S, Schiavone G, Turgut G, *Journal of Supercomputing*, **2008**, 43, 77–97.
[5] Hwang J J, Chow Y C, Anger F D, Lee C Y, *SIAM Journal of Computing*, **1989**, 18(2), 244–257.
[6] Wu M Y, Gajski D D, *IEEE Transactions on Parallel and Distributed Systems*, **1990**, 1(3), 330–343.
[7] Sih G C, Lee E A, *IEEE Transactions on Parallel and Distributed Systems*, **1993**, 4(2), 75–87.
[8] Kwok Y, Ahmad I, ACM *Computing Surveys*, **1999**, 31 (4), 407–471.
[9] Davidovic T, Crainic T, *Computer and Operation Research*, **2006**, 33, 2155–2177.
[10] Bonyadi M, Moghaddam M., *International Journal of Parallel Programming*, **2009**, 37 (5), 462-487.
[11] Tsuchiya T, Osada T, Kikuno T, *Journal of Microprocessors and Microsystems*, **1998**, 22(3–4), 197–207