LIST SCHEDLING ALGORITHMS FOR SOLVING IDENTICAL PARALLEL PROCESSOR IN MINIMIZING MAKESPAN

Nurul Izzati binti Muhammad & Syarifah Zyurina binti Nordin

Abstract

This study focuses on the task scheduling problem on identical parallel processors. We consider a non-preemptive task scheduling with an objective function of minimizing the makespan. Makespan is the maximum of completion time to entire set of tasks where $C_{max} = \max\{C_i, i = 1,2,\dots,n\}$.The standard assumptions of the task characteristic of this study are no delay schedule and no precedence constraints are required. Moreover, all tasks are ready at time zero and no due date or deadlines is specified. An arbitrary processing time of mathematical model of Mixed Integer Linear Programming (MILP) is considered to obtain the exact solution. We address three List Scheduling Algorithm, which are Shortest Processing Time, Longest Processing Time, and First Come First Served. The MILP model has been implemented using AIMMS 4.13 software package which uses CPLEX 12.6.2 as the solver for minimizing the makespan. The MILP gives the optimum result for each instance. A computational experiment is conducted to examine the effectiveness of the different size problem. The computational results show that all the proposed heuristics obtain good result with the gap between optimal solutions are less than 20% even for a large data set. Longest Processing Time (LPT) is the best List Scheduling heuristic method with maximum gap less than 2%.

Keywords : List Scheduling Algorithms; Identical parallel processor; Minimizing makespan.

Introduction

Scheduling is very common activity in industry and non-industry surroundings. Every day, meetings are scheduled, deadlines and work periods are set, maintenance and upgrade the operations are planned, lecture rooms are booked and others. Proper scheduling allows various activities or tasks to be completed in an organized manner. Scheduling plays an important role in management of a company. Scheduling is all about using mathematical techniques and heuristic method to allocate limited resources over time to perform a set of tasks in order to optimize objectives and achieve goals (Pinedo, 2002). According to Pinedo (2002), decision-making process is vital in procurement and production, in transportation and distribution, information processing and communication.

Scheduling is a method of assigning a number of tasks to process for processing and the scheduling program can be in serial (single) or parallel. In serial processing, the process runs in sequence while in parallel processing, it run in parallel. Therefore, parallel processing scheduling system takes less time to complete and more effective especially for problem with large volume of data. The purpose of the scheduling is to determine the allocation and the sequence of the operation to the target. The schedule needs to satisfy all the requirements and the constraints of the problem. However, the objective is to produce the best performance of scheduling system with efficiency policy (SyarifahZyurina, 2014).

In identical parallel processor scheduling problem, they have more than one processor are available for processing the tasks, they are identical and parallel. There are given tasks and each of these tasks has a processing time to be processed on the m identical parallel processor.

This study will focuses on the task scheduling problem on identical parallel processor. The standard assumptions of the task characteristic of this study are no delay schedule and no precedence constraints are required. Other than that, all tasks are ready at time zero and no due date or deadlines is specified. Moreover, objective function of identical parallel processor will focus on minimizing the makespan(C_{max}), where makespan is defined as $\max(C_1, C_2, \ldots, C_n)$, is the equivalent to the completion time of the last task to leave the system. The problem of identical parallel processor in minimizing makespan can be denoted as $P||C_{max}$.A minimum makespan usually implies a good utilization of the processor. In this study, we will use List Scheduling Algorithms to obtain the best heuristic methods and mathematical model of Mixed Integer Linear Programming (MILP) to obtain the exact or optimal solution.

Literature Review

Parallel processor scheduling has been a popular research field due to wide range of potential range of potential applications. The parallel processors can be can be grouped into three categories based on their operation, namely; identical, unrelated and uniform parallel processors. In identical parallel processors, a task can be processed at any one of the m processors with different processing time, Sun *et al.* (2003). In unrelated parallel processors, the task has to be performed only in the allocated particular processor Pinedo (2002), whereas, in uniform parallel processors the task can be done on any processor with equal processing times at all processor.

Identical parallel processor scheduling problem with task splitting and sequence is considered dependent setup times to minimize the maximum makespan within the set of all processor scheduling plans (Yalaoui and Chu, 2003). They develop two phase heuristic. First phase, they approached it as a single processor problem, transforming it into a Traveling Salesman Problem (TSP) and assign tasks to processors using Little's method (Little *et al.*, 1963). Second phase was created a feasible schedule for each processor, with the previously assigned tasks, which is improved taking advantage of the problems characteristics.

Nait*et al.* (2006) was used this method for the same problem, introduce a heuristic based on Linear Programming (LP) formulation to improve the approach Yalaoui and Chu (2003). According to Xing and Zhang (2000), they was studied about the task splitting property on an identical parallel processor scheduling problem with independent setup times to minimize the makespan, discussing cases with splitting properties and analysing a heuristic for this problem by extrapolating preemption properties. Another article by Mokotoff (2004), in the classical deterministic identical parallel machine problem, there are a number of independent jobs to be processed on a range of identical machines. Each job has to be carried out on one of the machines during a fixed processing time, without preemption. The problem of finding the schedule that optimizes the makespan is considered. Dynamic programming and branch and bound (B&B) techniques have been used to find optimal solutions.

Min and Cheng (1999) presented a kind of Genetic Algorithm (GA) based on processor code for minimizing the makespan in identical parallel processor scheduling. They demonstrate that the proposed GA is efficient ad fit for large scale problems and has advantage over heuristic procedure and simulated annealing method. Lee *et al.* (2006) propose a simulated annealing method to generate near optimal solutions for the minimization of makespan in identical parallel processor scheduling. With the help of computational analysis, they demonstrate that the proposed method is very accurate and outperforms the existing method. According to Hong *et al.* (2009), consider minimizing

makespan of identical parallel processor scheduling problems with mold constraints. In this kind of problems, tasks are non-preemptive with mold constraints and several identical processors are available. They proposed GA based approach to solve this problem. In parallel processor scheduling problem, no matter how many processors are involved, the number of workers at each processor may be ignored or assumed to be fixed and not taken into consideration. However, assigning more workers to work on the same task will decrease task completion time.

Methodology

Model description.Let consider the problem of scheduling is independent tasks on m identical parallel processors with availability constraints. The availability constraint means that some processors are not available for a certain period of time. Therefore, it cannot be used to process tasks. The objective function is to minimize the makespan.

In the modelling of this problem, the following standard assumptions are used :

a) All processor are identical and able to perform all operations.
b) Each processor can process only one task at any time.
c) Each part has only one, maybe complex, operation.
d) Preemption of a task on another processor is not allowed.
e) All tasks are available at time zero. However, some processors may not be available at that time.
f) Setup times are independent of task sequence and are included in the processing times.
g) The duration is known and constant. Therefore, off-line algorithm is used to solve the problem.

In this study, we will focus on method MILP which is defined the value of the variables are integer, the objective function and the constraints are linear. MILP problem is a mathematical optimization or feasibility program that can take exponential time to solve due to the combinatorial solving process. This method is the most power representation that usually used to formulate decision making problems under uncertainty in operation research and cooperative control.

Notations

The following notations are used for the problem under consideration.

i Index for task, $i = 1,2,3,\dots,n$

j Index for processor, $j = 1,2,3,\dots,m$

n Maximum number of task

m Maximum number of processor

p_i Processing time for task i

x_{ij} Assignment variable for task i on processor j

C_{max} Makespan

The MILP model for the problem $P||C_{max}$ can be written as follows :

Let $x_{ij} = \begin{cases} 1, & \text{if job } i \text{ is processed on processor } j \\ 0, & \text{otherwise} \end{cases}$

This variable to ensure each position on the list can hold only one task at the same time, and need to reveal that the first task should have setup time on the same processor. Setup time is required when a task is followed by another task from a different group and vice versa.

$$\text{Minimize} \quad C_{max} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ij}\, p_i \leq C_{max}, \quad j = 1,2,\ldots,m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1,2,\ldots,n \tag{3}$$

$$C_{max} \geq 0 \tag{4}$$

$$x_{ij} \in \{0,1\} \text{ for } i = 1,2,..n; \; j = 1,2,\ldots,m \tag{5}$$

In the above formulation, constraint (1) represents as the objective function, minimization of the makespan. Constraint (2) ensure that the sum of execution time for every task on processor j is less than or equal to C_{max}. Constraint (3) ensures that each task is assigned to only one of the m processors. Constraint (4) represent that makespan is greater or equal to zero. Constraint (5) ensure each position on the list can hold only one task at the same time, and need to reveal that the first task should have setup time on the same processor. Setup time is required when a task is followed by another task from a different group and vice versa.

List Scheduling Algorithms

In this study, the list scheduling algorithms that will be used are Shortest Processing Time (SPT), Longest Processing Time (LPT) and First Come First Served (FCFS).Shortest processing timeis a list of tasks that sequenced in ascending order of the processing time required at the processor, with the task requiring the least processing time at the processor scheduled first.Longest processing time is a list of tasks that sequenced in descending order of the processing time required at the processor, with the task requiring the longest processing time at the processor scheduled first. First come first servedis a list of tasks that sequenced in the general order in which they arrive at the processor. The tasks need to be done earlier will be at the front of the list.

Results And Analysis

The simulation data for the problem $P||C_{max}$ is generated as follows :

1. The number of independent tasks are n={10, 20, 30, 50}.
2. For every set of tasks, we use a different number of processors m ={2, 3, 5}.
   In total, there are 12 combinations of m and n.
3. For every combination of m by$n$, we generate 10 instances. Therefore, the total number of instances that we have are $12 \times 10 = 120$.
   We use an interval for the processing time where $p^{min}(i) = 0$and $p^{max}(i) = 20$.

Computational Results.

We now present our result of the SPT, LPT and FCFS algorithms compared with the optimal solutions. The result for the MILP model and algorithms will be presented as a gap (%) and can be calculated as follows :

$$Gap\ (\%) = \frac{C_{max,A} - C^*_{max}}{C^*_{max}} \times 100$$

where C^*_{max} is the optimum solution and $C_{max,A}$ is the value obtained when reached the specific time limit for SPT, LPT or FCFS.

Table 1 Result of the average makespan, $P||C_{max}$

Number of Processor m	Number of Task n	Average Makespan			
		MILP	SPT	LPT	FCFS
2	10	46.2	49.4	46.4	47.8
	20	99.9	104.9	100.1	100.8
	30	151	155.3	151.1	151.2
	50	256.2	261.1	256.2	256.4
3	10	31.2	36.4	31.6	33.4
	20	66.6	74.9	66.7	69
	30	100.9	107.5	100.9	102.4
	50	171	178.5	171	172.3
5	10	20.2	26.3	20.3	23.4
	20	40.4	48.4	40.6	44
	30	60.8	70.7	60.8	63.3
	50	102.7	111.5	102.7	105.4

Table 2 Gap (%) between list scheduling algorithms and optimal solution

Number of Processor m	Number of Task n	Gap (%)		
		SPT	LPT	FCFS
2	10	6.93	0.43	3.46
	20	5	0.2	0.9
	30	2.85	0.07	0.13
	50	1.91	0	0.08
3	10	16.67	1.28	7.05
	20	12.46	0.15	3.6
	30	6.54	0	1.49
	50	4.39	0	0.76
5	10	30.2	0.5	15.84
	20	19.8	0.5	8.91
	30	16.28	0	4.11
	50	8.57	0	2.63

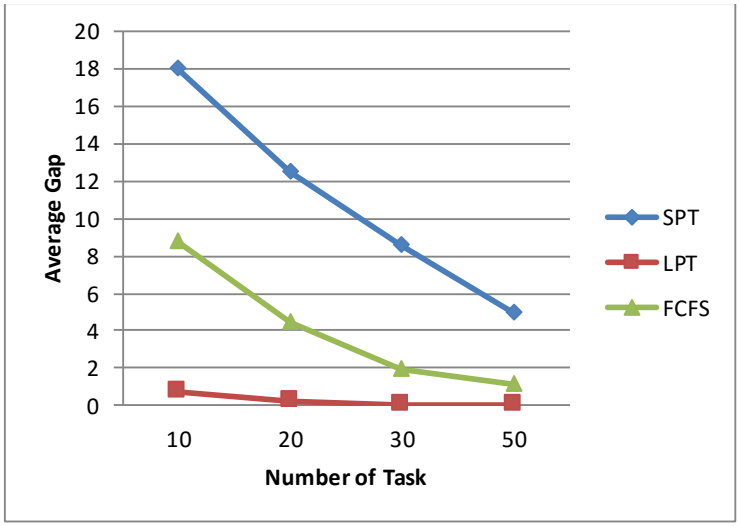

Figure 1 Average Gap against number of tasks for list scheduling algorithms

The objective is to evaluate the average gap of list scheduling algorithms when the data get larger. The experiment shows that the average gap for Figure 1 of SPT, LPT and FCFS are slightly decreased. The figure depicts the average gap for the larger number of tasks $n = 50$ becomes decrease for SPT, LPT and FCFS, this mean when the data greater than 50, it will near to optimal solution. Then, when the number of tasks increases, the average gap becomes decrease which means these three lists scheduling are best method when the data get larger and its stable method. The experiment shows that the gap has been shorter when more resources are allocated. However, longest processing time algorithms show the best heuristic method when the maximum gap between the MILP model and LPT around 2%. These results show us that as the number of tasks and processors increase, the list scheduling algorithms are getting better when the algorithms has more resources to allocate and increase the possibilities for tasks assignment.

Conclusion

In conclusion, we have achieved all of the objectives in this study. The computational results show that all the three list scheduling algorithms heuristics obtain good result with the gap of average between optimal solutions are less than 30% even for a large data set. Moreover, longest processing time algorithms show the best heuristic method when the maximum gap between the MILP model and LPT less than 2%. Then, the second heuristic method for solving identical parallel processor in minimizing makespan is FCFS with 16% of maximum gap.

Recommendations

For future research, there are some recommendations that can be suggested from this study. Firstly, the other list scheduling algorithms such as longest remain processing time (LRPT) or meta-heuristics algorithm for example Simulated Annealing, Tabu Search, and Genetic Algorithms can be used to solve identical parallel processor in minimizing makespan, $P||C_{max}$. Furthermore, other software like Matlab, LINGO/LINDO and CPLEX can be used to solve the MILP model and heuristic methods of the problem. Other than that, change the

standard assumptions of the task characteristic have delay schedule and precedence or preemtive constraints are required. Due date is specified for all tasks.

References

Hong TP, Sun PC, Jou SS (2009), Evolutionary computation for minimizingmakespan on identical parallel machine with mold constraints, WSEAS Trans Syst Control, 7(4);339-348.

Lee W. C, Wu C. C, Chen P (2006), A simulated annealing approach to makespanminimization on identical parallel machines. Int J AdvManuf Tech, 31:328-334.

Little, J. D., Murty, K. G., Sweeney, D. W., and Karel, C. (1963), An algorithm forthe traveling salesman problem, Operations research, 11(6):972–989.

Min L, Cheng W (1999), A genetic algorithm for minimizing the makespan in thecase of scheduling identical parallel machines, ArtifIntellEng, 13(4):399-403.

Mokotoff Ethel (2004), An exact algorithm for the identical parallel machinescheduling problem, European Journal of Operational Research, 152(2):758-769.

Nait, T. D., Yalaoui, F., Chu, C., and Amodeo, L. (2006), A linear programmingapproach for identical parallel machine scheduling with job splitting and sequence-dependent setup times, International Journal of Production Economics, 99(1):63–73.

Pinedo M. L. (2002), Scheduling : Theory, Algorithms, and Systems, Prentice Hall,2nd ed. 2002 edition.

Sun, H., Wang, G, (2003), Parallel machine earliness and tardiness scheduling withproportional weights, Computers & Operations Research, 30, 801-808.

SyarifahZyurinaNordin (2014), Lecture notes : Introduction of scheduling,Scheduling 2nd edition.

Xing, W. and Zhang, J. (2000), Parallel machine scheduling with splitting jobs,Discrete Applied Mathematics, 103(1):259–269.

Yalaoui, F. and Chu, C. (2003), An efficient heuristic approach for parallel machinescheduling with job splitting and sequence-dependent setup times, IIE Transactions, 35(2):183–190.