Jurnal Teknik Industri
Vol. 24, No. 1, February 2023, pp. 17-30
ISSN: 1978-1431 print | 2527-4112 online

Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan

Eka Pakpahan [a]*, Kenneth David Sumarna [a]
[a] Institut Teknologi Harapan Bangsa, Bandung, Indonesia
* Corresponding author: eka@ithb.ac.id

ARTICLE INFO

Article history
Received, June 14, 2022
Revised, November 4, 2022
Accepted, December 23, 2022
Available Online, March 27, 2023

Keywords
Scheduling
Heuristic Algorithms
Flexible machines
Job Shop
Makespan

ABSTRACT

This research examined the scheduling of jobs with multiple stages unto identical parallel machines to minimize the makespan. The work is motivated by a Flexible Manufacturing System case that produces various parts and has multiple machining centers. Early research towards this system proposed a stage-by-stage independent scheduling, resulting in a non-optimal solution. This study aimed to create a better solution for the system by developing a novel heuristic algorithm based on the classical longest processing time algorithm and simultaneously considering processing time for all stages when deciding the job sequencing and job-machine allocation. The algorithm is defined as Modified LPT for Multiple Identical Machine with Multi-process Capability (M-LPT MIMMPC). We performed a numerical experiment to assess the algorithm's performance by incorporating various cases. We concluded that the resulting makespans are always better than LPT's theoretical bound for parallel machine scheduling. In some cases, it successfully gave an optimal value. Although the experiment scope was still limited, the algorithm showed promising performance results.

1. Introduction

Delivering products at the right time begins with proper production scheduling. Scheduling is about deciding the sequence and resource allocation. This is especially true when considering multiple resources in our system [1]. The scheduling target varies from minimizing processing time, minimizing work in process and customer waiting time, and maximizing resource utilities and throughput. This research is motivated by a situation faced by a flexible manufacturing production system mentioned in [2] and [3]. During the planning stage, the management must decide on the production schedule. Multiple jobs must be scheduled, each of which needs several operation stages. These stages are associated with the feature creation sequence, making them strictly non-violated. Each job has different operation sequences. The system has multiple identical machining centers to process the jobs, each having a multi-process capability. The target of the system is minimizing the makespan. Research by [2] developed a mathematical model for this

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

system's scheduling problem. They solved the model through an analytical approach but suffered inefficiency in computational time. Research by [3] proposed an ant colony optimization method to create a schedule for the same system. Although metaheuristic implementation has improved the efficiency of solution finding, LPT is still used to decide the job processing sequence at each stage. Therefore, further research should be done to improve the quality of the solution while maintaining computational efficiency, which is the aim of this research.

Minimizing makespan on scheduling jobs unto identical parallel machines has been an interesting research topic for years. Implementing the heuristic method is often preferable because the problem is NP-Hard. The classic Longest Processing Time (LPT) algorithm is the most popular heuristic algorithm. This algorithm is known to generate a schedule with makespan satisfying error bound as much as $\frac{4}{3}-\frac{1}{3m}$ (with m = the number of parallel machines available) [1]. On the other side, in terms of computational efficiency, LPT has also shown high computational efficiency [4]. A continuous effort has been conducted to improve the performance of LPT, such as done by [5-8]. Research by [5] modified the LPT algorithm for scheduling jobs into two uniform parallel machines, and it passed the makespan performance proposed by the classical LPT. Research by [6] proposed the iterative use of the LPT algorithm and presented a novel approximation algorithm for minimizing makespan, whereas [7] combined LPT with local search iteration and effectively reduced makespan value in more than 80% of the cases studied. Another heuristic approach was proposed by [8] after careful analysis of the classical LPT performance. Another approach taken was by applying the approximation method [9], metaheuristics such as Particle Swarm Optimization [10] and Grey Wolf Optimizer [11], and simulation [12]. While this research shows promising performance, they only consider single-stage jobs.

Research on multiple stages is divided into flow shop (FS) and job shop (JS) situations. FS situation is when there are jobs that have similar stage sequence requirements. Each stage represents an operation on a particular machine. Research dealing with flow shop scheduling is done by [13-19]. Compared to FS, the JS situation happens when the jobs have various stage sequence requirements and mostly do not necessarily have the same number of stages [1].

The production system considered in this research falls into the job shop situation, which aims to schedule several jobs over some machines in which each job has a unique machine route. A review of the problem and solution algorithm for job shop scheduling was given by [20]. We mainly consider the type of job shop where multiple identical machines with multi-processing capability are available to service jobs, called Flexible Job Shop (FJS) [21]. The multi-capability embedded in each machine would affect the scheduling mechanism. The scheduler would have more flexibility when choosing which machine to process a job. Numerous papers were written in the context of FJS; among them, minimizing makespan is one of the most common targets [21]. The research can be divided into four groups. The first group considers the exact algorithm as a way to generate a schedule [22-24]. Research by [22] proposed constraint programming after analyzing the MILP (Mixed Integer Linear Programming) model of FJS, [23] adopted a branch and bound approach, and [24] introduced quantum computing-based optimization, which resulted in satisfactory performance and high practicability. The second group considers using the heuristic algorithm in dynamic FJS [25-27]. Research by [25] proposed a heuristic algorithm to accommodate variable processing time, [26] implemented greedy randomized adaptive search, and [27] combined local search and acceptance technique as an improved Jaya algorithm, thus resulting in solution quality improvement. The third

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

group used metaheuristic algorithms, including swarm optimization [28, 29], grey wolf optimization [30], and genetic algorithm [31]. The fourth group includes research considering simulation [32] and machine learning methods [33].

The position of this paper is in the second group, which considers the implementation of a heuristic algorithm on FJS, in this case, LPT. Our research aims to improve the research done by [2] and [3] by modifying the LPT algorithm to minimize makespan. Although the LPT algorithm has shown superior performance in minimizing makespan and delivering efficient computational time, research concerning LPT implementation in the FJS context is still rare. The following points summarize the contributions of this paper. First, we developed a novel algorithm based on the classic LPT algorithm to minimize makespan scheduling jobs with multiple stages unto multiple identical machines with multi-process capability. Second, we provided experiments and showed the effectiveness and computation efficiency of the proposed algorithm.

2. Methods

2.1 Problem Description and Assumptions List

We considered a production system described in [2] and [3]. There are J jobs, and each will be denoted by index j ($j = 1,2,\dots,J$). Each job has S stages to accomplish, and each stage will be denoted by index s ($s = 1,2,\dots,S$). The earlier stage must be completed before the next stage can begin. There are M available machines to process the jobs. Each machine has the same multi-process capability and can process any job allocated to it. The problem faced by the production planner is to allocate job-j, stage-s, on machine-m to obtain minimum makespan.

We introduce a list of mathematical notations and models for the problem.

Parameters:

$t_{j,s}$: Processing time of job-j stage-s

Variables:

CT_{max} : Makespan
CT_j : Completion time of job-j
$CT_{j,s,m}$: Completion time of job-j, stage-s, at machine-m
$ST_{j,s,m}$: Start time of job-j, stage-s, at machine-m

Decision Variables:

$X_{j,s,m}$: 1, if job-j, stage-s is assigned to machine-m; 0, otherwise

Objective function:

$$\text{Min } CT_{max} \tag{1}$$

Constraints:

$$CT_{max} \geq CT_j, \quad \forall j \tag{2}$$

$$CT_j \geq \sum_{m=1}^{M} CT_{j,s=S,m}, \quad \forall j \tag{3}$$

$$CT_{j,s,m} \geq \left(ST_{j,s,m} + t_{j,s}\right) - \left(1 - X_{j,s,m}\right)L, \quad \forall j, \forall s, \forall m \tag{4}$$

$$\sum_{m=1}^{M} ST_{j,s,m} \geq \sum_{m=1}^{M} CT_{j,(s-1),m}, \quad \forall j, \forall s \tag{5}$$

$$CT_{j,s=0,m} = 0, \forall j, \forall m \tag{6}$$

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

$$ST_{j,s,m} + CT_{j,s,m} \leq X_{j,s,m}.L \quad (7)$$

$$X_{j,s,m} \in \{0,1\}, \forall j, \forall s, \forall m \quad (8)$$

$$\sum_{m=1}^{M} X_{j,s,m} = 1, \forall j, \forall s \quad (9)$$

$$ST_{j,s,m} \geq 0, \forall j, \forall s, \forall m \quad (10)$$

$$CT_{j,s,m} \geq 0, \forall j, \forall s, \forall m \quad (11)$$

$$CT_j \geq 0, \forall j \quad (12)$$

$$CT_{max} \geq 0 \quad (13)$$

Equation (1) defines the problem's objective function, minimizing the schedule's makespan. Equation (2) defines makespan as the maximum value of a job's completion time. Equation (3) states that a job's completion time depends on the completion time of its last stage process. Equation (4) defines the completion time of job$-j$, stage-s at machine-m. It depends on the starting time, processing time, and machine allocation. Equation (5) states that a job for a particular stage cannot start before its predecessor stage has finished. Equation (6) complements Equation (5) by providing the completion time value for job-j, stage-0, which means that the starting time for the first stage on each job is always larger or equal to 0. Equation (7) ensured that each stage's starting and completion times occurred in the same machine. Equation (8) defines the possible value of $X_{j,s,m}$, which is either 0 or 1. Equation (9) ensures that job-j stage-s is allocated in only one machine. Equation (10) – (13) defines the non-negativity value for variables starting time, completion time of job-j, stage-s at machine-m, the completion time of job-j, and the makespan.

Three assumptions are applied when dealing with the scheduling problem for this system. The first is that machines are assumed to be available at any time. The second is that the setup time is included in the processing time, and the last is that material handling processes are done instantly.

2.2 Algorithm Development

When dealing with single-stage scheduling, the LPT algorithm derives a near-optimal solution for minimizing makespan [1],[4-8]. It is known to have two general steps. The first is arranging the job sequence, and the second is allocating them to machines [1]. The arrangement of job sequence is made by implementing the longest processing time rule, where jobs are ordered based on their processing time in non-increasing order. Afterward, the allocation is done by prioritizing machines with the lowest load. However, when we deal with the case of jobs with multiple stages, implementing LPT would potentially create overlaps between stages, thus leading to an infeasible solution. To avoid this problem, we proposed a new algorithm based on modification towards the classic LPT algorithm. We will mention the algorithm as Modified LPT for Multiple Identical Machine with Multi-process Capability (M-LPT MIMMPC).

The M-LPT MIMMPC is divided into three sub-algorithms. The first and the second sub-algorithms aim to form surrogate lists for the processing times used as input for the third sub-algorithm. Details of each step are explained in Table 1, Table 2, and Table 3.

Given the list generated by sub-algorithm 1 (Table 1) and sub-algorithm 2 (Table 2), sub-algorithm 3 (Table 3) would arrange the list in non-increasing order. The ordered

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

list would be used to decide the job allocation sequence. We also added a comparison mechanism before deciding the job starting time on the selected machine. The starting time of a job should never be earlier than its predecessor's completion time. There are two schedules created by sub-algorithm 3, but in the end, one was selected based on the minimum makespan.

Table 1. M-LPT MIMMPC Sub-Algorithm A (Forward Comparison)

Input	:	Number of jobs (***J***), number of stages (***S***), and processing time ($\boldsymbol{t_{j,s}}$)
Step 1	:	Set $j = 1$. Continue to Step 2.
Step 2	:	Create an initial surrogate list for processing time. For $j = 1$ to s, set $t^*_{j,s} = t_{j,s}$. Continue to Step 3.
Step 3	:	Set $s = 1$. Continue to Step 4.
Step 4	:	Decide whether $t^*_{j,s} > t^*_{j,(s+1)}$. If yes, continue to Step 5; otherwise, continue to Step 6.
Step 5	:	Check whether $s + 1 = S$. If yes, go to Step 9; otherwise, set $s = s + 1$, then go back to Step 4.
Step 6	:	Set $t^*_{j,s} = t^*_{j,s} + t^*_{j,(s+1)}$; that is, merging the two-processing time. Continue to Step 7.
Step 7	:	Check whether $s + 1 = S$. If yes, go to Step 9; otherwise, continue to Step 8.
Step 8	:	Update the initial surrogate list by eliminating the $(s + 1)$ column. It would reduce the length of the list to $(S - 1)$ and change each value of $t^*_{j,(s+1)}$ by $t^*_{j,(s+2)}$. Go back to Step 3.
Step 9	:	Check whether $j = J$. If yes, go to Step 10; otherwise, set $j = j + 1$, then go back to Step 2.
Step 10	:	Define the initial surrogate list as the final surrogate list$_{-1}$. STOP
Output	:	Final surrogate list$_{-1}$ for Processing Time

Table 2. M-LPT MIMMPC Sub-Algorithm B (Backward Comparison)

Input	:	Number of jobs (***J***), number of stages (***S***), and processing time ($\boldsymbol{t_{j,s}}$)
Step 1	:	Set $i = 1$. Continue to Step 2.
Step 2	:	Create an initial surrogate list for processing time. For $j = 1$ to J, set $t^*_{j,s} = t_{j,s}$. Continue to Step 3.
Step 3	:	Set $s = S$. Continue to Step 4.
Step 4	:	Decide whether $t^*_{j,s} > t^*_{j(s-1)}$. If yes, continue to Step 5; otherwise, continue to Step 6.
Step 5	:	Check whether $s - 1 = 1$. If yes, go to Step 9; otherwise, set $s = s - 1$, then go back to Step 4.
Step 6	:	Set $t^*_{j,s} = t^*_{j,s} + t^*_{j,(s-1)}$; that is, merging the two-processing time. Continue to Step 7.
Step 7	:	Check whether $s - 1 = 1$. If yes, go to Step 9; otherwise, continue to Step 8.
Step 8	:	Update the initial surrogate list by eliminating the $(s - 1)$ column. It would reduce the length of the list to $(s - 1)$ and change each value of $t^*_{j,(s-1)}$ by $t^*_{j,(s-2)}$. Go back to Step 3.
Step 9	:	Check whether $j = J$ If yes, go to Step 10; otherwise, set $j = j + 1$, then go back to Step 2.
Step 10	:	Define the initial surrogate list as the final surrogate list$_{-2}$. STOP
Output	:	Final Surrogate List$_{-2}$ for Processing Time

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

The M-LPT MIMMPC algorithm was built into Matlab programming language. The program is then run in the Windows OS setting. The computer used during the experiment has the following specifications: Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz.

Table 3. M-LPT MIMMPC Sub-Algorithm C (Allocating Job unto Machines)

Input	:	The number of surrogate lists (**2**); the final surrogate list$_{_\mathbf{1}}$ and the final surrogate list$_{_\mathbf{2}}$; allocated job as an empty list; the number of machines ($\boldsymbol{M}$); the initial load on each machine ($\boldsymbol{B_m = 0, m = 1, 2, \dots, M}$) and the machine with the lowest load ($\mathbf{argmin}_{\boldsymbol{m}} \boldsymbol{B_m}$) identified as $\boldsymbol{m^*}$
Step 1	:	Set $k = 1$. Continue to Step 2.
Step 2	:	Implement LPT rules (non-increasing order) to all $t^*_{j,s}$ on the Final surrogate list$_{_k}$ defined it as an Ordered surrogate list$_{_k}$. Continue to Step 3.
Step 3	:	Set $l = 1$. Continue to Step 4.
Step 4	:	For element-l on the Ordered surrogate list$_{_k}$, identify the value of $t^*_{j,s}$, and allocate element-l to machine-m^*. Continue to Step 5.
Step 5	:	Check whether the selected $t^*_{j,s}$ has a predecessor at the allocated job list. If yes, continue to Step 6; otherwise, continue to Step 7.
Step 6	:	Identify the completion time of the job's predecessor ($CT_{j,(s-1),m}$) Set start time (ST_{j,s,m^*}) and completion time (CT_{j,s,m^*}) for element-l and update the load on the selected machine (B_{m^*}) $ST_{j,s,m^*} = CT_{j(s-1),m^*}$; $CT_{j,s,m^*} = ST_{j,s,m^*} + t^*_{j,s}$ and $B_m = CT_{j,s,m^*}$ Continue to Step 8.
Step 7	:	Identify the load on the selected machine (B_{m^*}) Set start time (ST_{j,s,m^*}) and completion time (CT_{j,s,m^*}) for element-l and update the load on the selected machine (B_{m^*}) $ST_{j,s,m^*} = B_m$; $CT_{j,s,m^*} = ST_{j,s,m^*} + t^*_{j,s}$ and $B_m = CT_{j,s,m^*}$. Continue to Step 8.
Step 8	:	Update the identity of the machine with the lowest load $\underset{m}{\text{argmin}}\, B_m$ or (m^*). Continue to Step 9.
Step 9	:	Update the allocated job list by including $t^*_{j,s}$. Continue to Step 10.
Step 10	:	Check whether all elements in the surrogate list$_{_k}$ has been allocated If yes, continue to Step 11; otherwise, set $l = l + 1$ and go back to Step 4
Step 11	:	Determine the makespan of the schedule $MS_k = \max(B_m)$; go to Step 12.
Step 12	:	Check whether $k = 2$ If yes, continue to Step 13; otherwise, set $k = k + 1$, then go back to Step 2
Step 13	:	Find the schedule with minimum makespan and the associated$-k$; $CT_{max} = min\, MS_k$; $\underset{k}{\text{argmin}}\ MS_k$
Output	:	Allocation of jobs unto machines (schedule) and its associated makespan

To ensure that the algorithm performed as intended, we created small cases and traced the solution's execution of steps and logic. Afterward, an experiment was conducted to assess the algorithm's performance regarding solution quality and computational time. The cases used in the experiment are designed intentionally to investigate the performance of the M-LPT MIMMPC algorithm given various situations. The cases are explained next.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

In case 1 (10 jobs with 2 stages), we created two sub-cases for this setting. The first sub-case is when $t_{j,1} > t_{j,2}$ (See Case 1a, in Table 4) and the second sub-case is when $t_{j,1} < t_{j,2}$ (see Case 1b, in Table 5.)

Table 4. Case 1a (10 jobs - 2 stages) where $t_{j,1} > t_{j,2}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	100	98	96	95	92	90	88	86	84	82
	Stage 2	90	88	86	84	82	80	78	76	74	72

Table 5. Case 1b (10 jobs - 2 stages) where $t_{j,1} < t_{j,2}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	100	98	96	95	92	90	88	86	84	82

In case 2 (10 jobs with 3 stages), we created three sub-cases for this setting. The first sub-case is when $t_{j,1} > t_{j,2} > t_{j,3}$ (see Case 2a, in Table 6), the second sub-case is when $t_{j,1} < t_{j,2}$ and $t_{j,2} > t_{j,3}$ (see Case 2b, in Table 7), and the third sub-case is when $t_{j,2} < t_{j,3}$ and $t_{j,1} < (t_{j,2} + t_{j,3})$ (see Case 2c, in Table 8).

Table 6. Case 2a (10 jobs - 3 stages) where $t_{j,1} > t_{j,2} > t_{j,3}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	100	98	96	95	92	90	88	86	84	82
	Stage 2	90	88	86	84	82	80	78	76	74	72
	Stage 3	50	48	46	44	42	40	38	36	34	32

Table 7. Case 2b (10 jobs - 3 stages) where $t_{j,1} < t_{j,2}$ and $t_{j,2} > t_{j,3}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	100	98	96	95	92	90	88	86	84	82
	Stage 3	50	48	46	44	42	40	38	36	34	32

Table 8. Case 2c (10 jobs - 3 stages) where $t_{j,2} < t_{j,3}$ and $t_{j,1} < (t_{j,2} + t_{j,3})$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	50	48	46	44	42	40	38	36	34	32
	Stage 3	100	98	96	95	92	90	88	86	84	82

In case 3 (10 jobs with 4 stages), we created four sub-cases for this setting. The first sub-case is when $t_{j,1} > t_{j,2} > t_{j,3} > t_{j,4}$ (see Case 3a, in Table 9), the second sub-case is when $t_{j,1} < t_{j,2}$; $t_{j,2} > t_{j,3}$ and $t_{j,3} > t_{j,4}$ (see Case 3b, in Table 10), the third sub-case is when $t_{j,1} < t_{j,2}$; $t_{j,2} > t_{j,3}$; $t_{j,3} < t_{j,4}$ and $(t_{j,1} + t_{j,2}) > (t_{j,3} + t_{j,4})$, (see Case 3c, on Table 11), the last sub-case is when $t_{j,1} > t_{j,2} > t_{j,3}$ and $t_{j,3} < t_{j,4}$ (see Case 3d, in Table 12).

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

Table 9. Case 3a (10 jobs - 4 stages) where $t_{j,1} > t_{j,2} > t_{j,3} > t_{j,4}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	100	98	96	95	92	90	88	86	84	82
	Stage 2	90	88	86	84	82	80	78	76	74	72
	Stage 3	50	48	46	44	42	40	38	36	34	32
	Stage 4	40	38	36	34	32	30	28	26	24	22

Table 10. Case 3b (10 jobs - 4 stages) where $t_{j,1} < t_{j,2}$; $t_{j,2} > t_{j,3}$ and $t_{j,3} > t_{j,4}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	100	98	96	95	92	90	88	86	84	82
	Stage 3	50	48	46	44	42	40	38	36	34	32
	Stage 4	40	38	36	34	32	30	28	26	24	22

Table 11. Case 3c (10 jobs - 4 stages) where $t_{j,1} < t_{j,2}$; $t_{j,2} > t_{j,3}$; $t_{j,3} < t_{j,4}$ and $(t_{j,1} + t_{j,2}) > (t_{j,3} + t_{j,4})$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	100	98	96	95	92	90	88	86	84	82
	Stage 3	40	38	36	34	32	30	28	26	24	22
	Stage 4	50	48	46	44	42	40	38	36	34	32

Table 12. Case 3d (10 jobs - 4 stages) where $t_{j,1} > t_{j,2} > t_{j,3}$ and $t_{j,3} < t_{j,4}$

Job		1	2	3	4	5	6	7	8	9	10
Processing time (minutes)	Stage 1	90	88	86	84	82	80	78	76	74	72
	Stage 2	50	48	46	44	42	40	38	36	34	32
	Stage 3	40	38	36	34	32	30	28	26	24	22
	Stage 4	100	98	96	95	92	90	88	86	84	82

When a particular optimization method or algorithm produces MS^*, the performance of the optimization method or algorithm can be calculated through its error. The error is defined as $\frac{MS^*}{MS^i}$ [1]. As we can see, reaching an error value equal to 1 means that the algorithm can reach the ideal solution. We use this value as a performance measure for the M-LPT MIMMPC and compare it to the error produced by the classic LPT algorithm.

3. Results and Discussion

3.1 Experiment Result

3.1.1 Case 1 (10 jobs with two stages)

The execution of M-LPT MIMMPC sub-algorithms 1 and 2 in Case 1a would result in the separation of each $t_{j,1}$ and $t_{j,2}$. Performing the LPT rule for all processing times in Case 1a would not result in stage violation. On the contrary, $t_{j,1}$ and $t_{j,2}$ for each-i in Case 1b would be merged and considered as one value (surrogate value) during LPT implementation. Merging the processing time would translate to processing the job simultaneously for both stages. A summary of M-LPT MIMMPC performance for Case 1 is given in Table 13.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

Table 13. Makespan, Error, and Calculation Time for Case 1.

Case	Number of Machines	MS^* (minutes)	MS^i (minutes)	Error	Error Bound	Calculation Time (Second)
1a	2	860	860	1.00	1.17	0.76
	3	598	573.33	1.04	1.22	0.64
	4	432	430	1.00	1.25	0.53
1b	2	862	860	1.01	1.17	0.58
	3	672	573.33	1.17	1.22	0.50
	4	510	430	1.19	1.25	0.56

3.1.2 Case 2 (10 jobs with three stages)

As in Case 1a, executing sub-algorithms 1 and 2 of M-LPT MIMMPC towards Case 2a would result in the separation of $t_{j,1}$, $t_{j,2}$ and $t_{j,3}$. Furthermore, no surrogate value would be applied for each of them. Case 2b would result in the merging of $t_{j,1}$ and $t_{j,2}$ and the separation of $t_{j,3}$. Case 2c would result in the merging $t_{j,1}$, $t_{j,2}$ and $t_{j,3}$. The summary of M-LPT MIMMPC performance for Case 2 is given in Table 14.

Table 14. Makespan, Error, and Calculation Time for Case 2.

Case	Number of Machines	MS^* (minutes)	MS^i (minutes)	Error	Error Bound	Calculation Time (Second)
2a	2	1066	1065	1.01	1.17	0.66
	3	714	710	1.01	1.22	0.66
	4	548	532.5	1.03	1.25	0.73
2b	2	1066	1065	1.01	1.17	0.63
	3	728	710	1.03	1.22	0.57
	4	544	532.5	1.02	1.25	0.46
2c	2	1068	1065	1.01	1.17	0.57
	3	828	710	1.17	1.22	0.57
	4	630	532.5	1.18	1.25	0.48

3.1.3 Case 3: 10 jobs with four stages

Following the pattern of Case 1a and 2a, Case 3a would result in the separation of $t_{j,1}$, $t_{j,2}$, $t_{j,3}$ and $t_{j,4}$ during LPT implementation. Case 3b would result in the merging of $t_{j,1}$and $t_{j,2}$ and separation of each $t_{j,3}$ and $t_{j,4}$. Case 3c would result in the merging of $t_{j,1}$ and $t_{j,2}$ as well as the merging of $t_{j,3}$ and $t_{j,4}$. In the last case, Case 3d, the sub-algorithms 1 and 2 of M-LPT MIMMPC would result in the merging of $t_{j,1}$, $t_{j,2}$, j, and $t_{j,4}$. The summary of M-LPT MIMMPC performance for Case 3 is given in Table 15.

3.2 Discussions

Given the three cases explained earlier, we summarize the performance of the M-LPT MIMMPC algorithm. The average error value is depicted in Fig 1, and the computational time is in Fig 2. Each line on the graph represented a data series for each case.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

Table 15. Makespan, Error, and Calculation Time for Case 3.

Case	Number of Machines	MS^* (minutes)	MS^i (minutes)	Error	Error Bound	Calculation Time (Second)
3a	2	1220	1220	1.00	1.17	0.90
	3	822	813.33	1.01	1.22	0.97
	4	612	610	1.01	1.25	0.82
3b	2	1220	1220	1.00	1.17	0.70
	3	820	813.33	1.01	1.22	0.82
	4	616	610	1.01	1.25	0.62
3c	2	1220	1220	1.00	1.17	0.83
	3	818	813.33	1.01	1.22	0.71
	4	638	610	1.05	1.25	0.73
3d	2	1224	1220	1.01	1.17	0.67
	3	944	813.33	1.16	1.22	0.56
	4	720	610	1.18	1.25	0.59

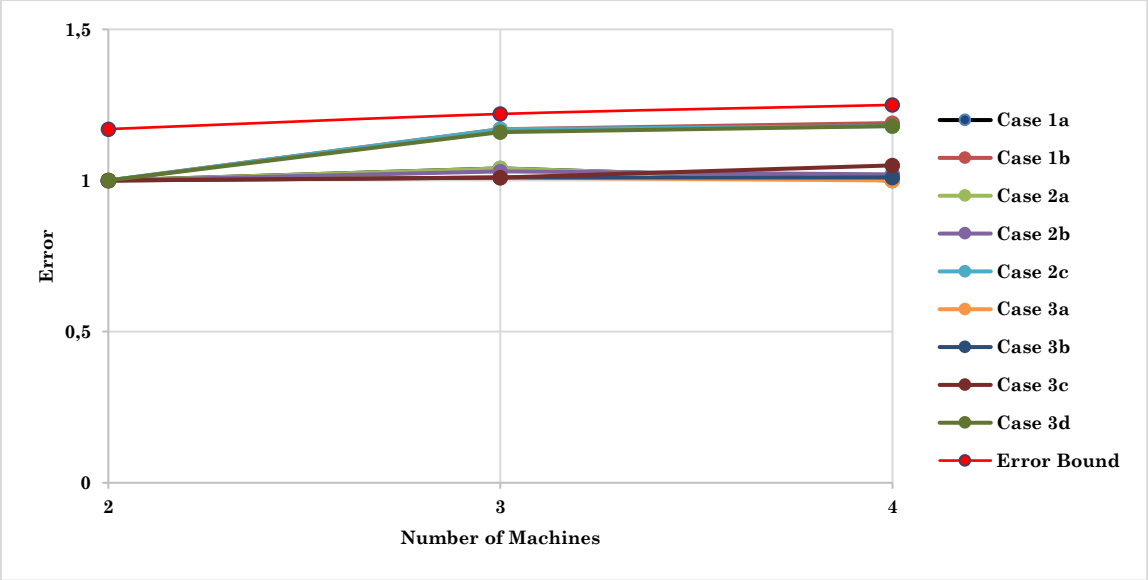

Fig 1. Average error value of M-LPT MIMMPC algorithm in various cases.

Fig 1 Shows that the error value in each case is always mapped below the error bound. When looking at this, we know that the proposed algorithm always produces a better solution than the error bound of the classical LPT algorithm. Another interesting result is that the algorithm can match the outstanding makespan value for most of the two machines scheduling settings. The worst performance of the algorithm was achieved primarily on the four machines scheduling settings at the error value of 1.19. It is shown in the data that the higher the number of machines, the larger the error value achieved by the proposed algorithm. These results mimic the pattern shown on the error bound derived from LPT [1]. It is safe to say that the proposed algorithm behaves like its preceding algorithm.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

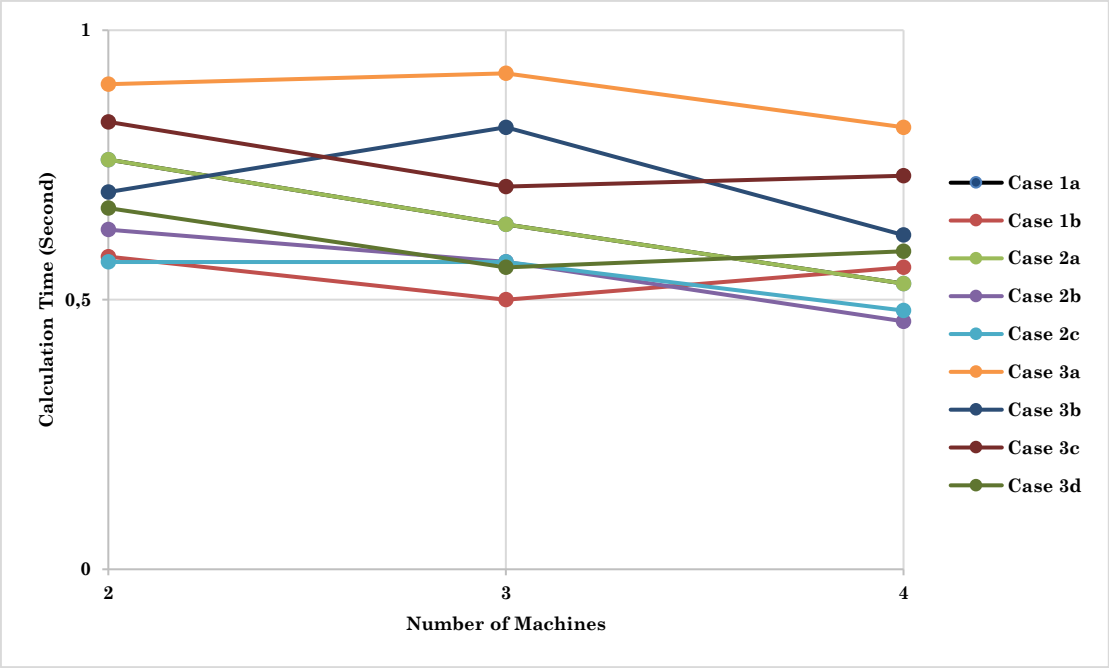

Fig 2. The average calculation time of the M-LPT MIMMPC algorithm in various cases.

Another critical performance measure to consider for practicality issues is computational efficiency. Fig 2 The proposed algorithm's computational time never exceeds one second in all cases. The general pattern found in the experiment result shows that the more stages the case considered, the higher the computational time needed. At specific stages, the computational time is determined by the number of processing times (or surrogate processing time) the algorithm must sort and allocate to machines. The less the merging process, the higher the number of jobs to be sorted and allocated, thus resulting in higher computational time.

While we only experimented on multiple-stage cases, the proposed algorithm can also be implemented in a single-stage setting. It would lead the M-LPT MIMMPC algorithm back to the classical LPT because the result of sub-algorithm 1 and sub-algorithm 2 is similar to the processing time of each job. This general form would be beneficial in practical terms.

4. Conclusion and Further Research

The experiment shows a promising performance of the proposed algorithm (M-LPT MIMMPC). It is proved that the algorithm can produce a better makespan than classical LPT. In several cases, it can even match the optimal value. The algorithm is also efficient regarding computational time and has general applicability. It can be used in a single-stage setting or multiple-stage setting. Combining these qualities shows effectiveness and efficiency, which is essential in real-world applicability. Even so, the experiment conducted in this research is still limited. We only consider 10 jobs, 2 to 4 stages, and 2 to 4 machine settings. Experimenting on a larger case is still needed. Future research should explore this and try to derive the exact error value given by the algorithm analytically.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

Declarations

Author contribution: We declare that both authors contributed equally to this paper and approved the final paper.
Funding statement: No funding was received for this work.
Conflict of interest: The authors declare no conflict of interest.
Additional information: No additional information is available for this paper.

Acknowledgment

We thank Institut Teknologi Harapan Bangsa for their support in allowing this research to be conducted. We also want to thank the anonymous reviewers for their insightful comments and suggestions to improve this paper.

References

[1] K. R. Baker and D. Trietsch, *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[2] A. Setiawan, R. Wangsaputra, Y. Y. Martawirya, and A. H. Halim, "A production scheduling model considering cutting tools for an FMS to minimize makespan," 2015: Asia Pacific Industrial Engineering and Management Society: Ho Chi Minh International Univ., 2015.

[3] E. K. A. Pakpahan, S. Kristina, and A. Setiawan, "Proposed algorithm to improve job shop production scheduling using ant colony optimization method," *IOP Conference Series: Materials Science and Engineering,* vol. 277, no. 1, p. 012050, 2017. https://doi.org/10.1088/1757-899X/277/1/012050.

[4] D. Laha and D. K. Behera, "A comprehensive review and evaluation of LPT, MULTIFIT, COMBINE and LISTFIT for scheduling identical parallel machines," *International Journal of Information and Communication Technology,* vol. 11, no. 2, pp. 151-165, 2017. https://doi.org/10.1504/IJICT.2017.086246.

[5] C. Koulamas and G. J. Kyparisis, "A modified LPT algorithm for the two uniform parallel machine makespan minimization problem," *European Journal of Operational Research,* vol. 196, no. 1, pp. 61-68, 2009. https://doi.org/10.1016/j.ejor.2008.02.008.

[6] H. habiba, A. Hassam, Z. Sari, C. M. Amine, and T. Souad, "Minimizing Makespan on Identical Parallel Machines," in *2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*, 2019, vol. 1, pp. 1-6. https://doi.org/10.1109/ICAAID.2019.8934993.

[7] D. De Giovanni, J. C. Ho, G. Paletta, and A. J. Ruiz-Torres, "Heuristics for Scheduling Uniform Machines," 2018, vol. 2, pp. 937-939: IMECS. https://www.iaeng.org/publication/IMECS2018/IMECS2018_pp937-939.pdf.

[8] F. Della Croce and R. Scatamacchia, "The Longest Processing Time rule for identical parallel machines revisited," *Journal of Scheduling,* vol. 23, no. 2, pp. 163-176, 2020. https://doi.org/10.1007/s10951-018-0597-6.

[9] L. Ghalami and D. Grosu, "Scheduling parallel identical machines to minimize makespan:A parallel approximation algorithm," *Journal of Parallel and Distributed Computing,* vol. 133, pp. 221-231, 2019. https://doi.org/10.1016/j.jpdc.2018.05.008.

[10] I. Alharkan, M. Saleh, M. A. Ghaleb, H. Kaid, A. Farhan, and A. Almarfadi, "Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server," *Journal of King Saud*

University - Engineering Sciences, vol. 32, no. 5, pp. 330-338, 2020. https://doi.org/10.1016/j.jksues.2019.03.006.

[11] S. Kamaraj and M. Saravanan, "Optimisation of identical parallel machine scheduling problem," *International Journal of Rapid Manufacturing,* vol. 8, no. 1-2, pp. 123-132, 2018. https://doi.org/10.1504/IJRAPIDM.2019.097033.

[12] F. Yu, P. Wen, and S. Yi, "A multi-agent scheduling problem for two identical parallel machines to minimize total tardiness time and makespan," *Advances in Mechanical Engineering,* vol. 10, no. 2, p. 1687814018756103, 2018. https://doi.org/10.1177/1687814018756103.

[13] M. Khatami, A. Salehipour, and F. J. Hwang, "Makespan minimization for the m-machine ordered flow shop scheduling problem," *Computers & Operations Research,* vol. 111, pp. 400-414, 2019. https://doi.org/10.1016/j.cor.2019.06.012.

[14] I. Amallynda, "The Discrete Particle Swarm Optimization Algorithms for Permutation Flowshop Scheduling Problem," *Jurnal Teknik Industri,* vol. 20, no. 2, pp. 105-116, 2019. https://doi.org/10.22219/JTIUMM.Vol20.No2.105-116.

[15] I. Amallynda and B. Hutama, "The Moth-Flame Optimization Algorithm for Flow Shop Scheduling Problem with Travel Time," *Jurnal Teknik Industri,* vol. 22, no. 2, pp. 224-235, 2021. https://doi.org/10.22219/JTIUMM.Vol22.No2.224-235.

[16] D. Marsetiya Utama, "An Effective Hybrid Sine Cosine Algorithm to Minimize Carbon Emission on Flow-shop Scheduling Sequence Dependent Setup," *Jurnal Teknik Industri,* vol. 20, no. 1, pp. 62-72, 2019. https://doi.org/10.22219/JTIUMM.Vol20.No1.62-72.

[17] A. N. A. A. K. Jabari and A. Hasan, "Energy-Aware Scheduling in Hybrid Flow Shop using Firefly Algorithm," *Jurnal Teknik Industri,* vol. 22, no. 1, pp. 18-30, 2021. https://doi.org/10.22219/JTIUMM.Vol22.No1.18-30.

[18] N. Dridi and A. O. Bedhief, "Minimizing makespan in a three-stage hybrid flow shop with dedicated machines," *International Journal of Industrial Engineering Computations,* vol. 10, no. 2, pp. 161-176, 2019. http://dx.doi.org/10.5267/j.ijiec.2018.10.001.

[19] J. Zhu, H. Wang, and T. Zhang, "A Deep Reinforcement Learning Approach to the Flexible Flowshop Scheduling Problem with Makespan Minimization," in *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, 2020, pp. 1220-1225. https://doi.org/10.1109/DDCLS49620.2020.9275080.

[20] C. Cebi, E. Atac, and O. K. Sahingoz, "Job Shop Scheduling Problem and Solution Algorithms: A Review," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1-7. https://doi.org/10.1109/ICCCNT49239.2020.9225581.

[21] J. Xie, L. Gao, K. Peng, X. Li, and H. Li, "Review on flexible job shop scheduling," *IET Collaborative Intelligent Manufacturing,* vol. 1, no. 3, pp. 67-77, 2019. https://doi.org/10.1049/iet-cim.2018.0009.

[22] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Computers & Industrial Engineering,* vol. 142, p. 106347, 2020. https://doi.org/10.1016/j.cie.2020.106347.

[23] C. Soto, B. Dorronsoro, H. Fraire, L. Cruz-Reyes, C. Gomez-Santillan, and N. Rangel, "Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm," *Swarm and Evolutionary Computation,* vol. 53, p. 100632, 2020. https://doi.org/10.1016/j.swevo.2019.100632.

[24] B. Denkena, F. Schinkel, J. Pirnay, and S. Wilmsmeier, "Quantum algorithms for process parallel flexible job shop scheduling," *CIRP Journal of Manufacturing*

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30

Science and Technology, vol. 33, pp. 100-114, 2021. https://doi.org/10.1016/j.cirpj.2021.03.006.

[25] M. Shahgholi Zadeh, Y. Katebi, and A. Doniavi, "A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times," *International Journal of Production Research,* vol. 57, no. 10, pp. 3020-3035, 2019. https://doi.org/10.1080/00207543.2018.1524165.

[26] A. Baykasoğlu, F. S. Madenoğlu, and A. Hamzadayı, "Greedy randomized adaptive search for dynamic flexible job-shop scheduling," *Journal of Manufacturing Systems,* vol. 56, pp. 425-451, 2020. https://doi.org/10.1016/j.jmsy.2020.06.005.

[27] R. H. Caldeira and A. Gnanavelbabu, "Solving the flexible job shop scheduling problem using an improved Jaya algorithm," *Computers & Industrial Engineering,* vol. 137, p. 106064, 2019. https://doi.org/10.1016/j.cie.2019.106064.

[28] R. Zarrouk, I. E. Bennour, and A. Jemai, "A two-level particle swarm optimization algorithm for the flexible job shop scheduling problem," *Swarm Intelligence,* vol. 13, no. 2, pp. 145-168, 2019. https://doi.org/10.1007/s11721-019-00167-w.

[29] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 4, pp. 904-916, 2019. https://doi.org/10.1109/JAS.2019.1911540.

[30] T. Jiang and C. Zhang, "Application of Grey Wolf Optimization for Solving Combinatorial Problems: Job Shop and Flexible Job Shop Scheduling Cases," *IEEE Access,* vol. 6, pp. 26231-26240, 2018. https://doi.org/10.1109/ACCESS.2018.2833552.

[31] G. Zhang, L. Zhang, X. Song, Y. Wang, and C. Zhou, "A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem," *Cluster Computing,* vol. 22, no. 5, pp. 11561-11572, 2019. https://doi.org/10.1007/s10586-017-1420-4.

[32] R. H. Caldeira and A. Gnanavelbabu, "A simheuristic approach for the flexible job shop scheduling problem with stochastic processing times," *SIMULATION,* vol. 97, no. 3, pp. 215-236, 2020. https://doi.org/10.1177/0037549720968891.

[33] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Computers & Industrial Engineering,* vol. 149, p. 106778, 2020. https://doi.org/10.1016/j.cie.2020.106778.

Please cite this article as: Pakpahan, E., & Sumarna, K. D. (2023). Novel Heuristic Algorithm for Flexible Job Shop Scheduling based on the Longest Processing Time Rules to Minimize Makespan. Jurnal Teknik Industri, 24(1), 17–30. https://doi.org/10.22219/JTIUMM.Vol24.No1.17-30