Received: August 13, 2021. Revised: September 13, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Parallel Permutation Flow-Shop Scheduling Model by Using a Two-Step Evolutionary Algorithm to Minimize Intermediate Storage with Tolerable Maximum Completion Time

Purba Daru Kusuma[1]* **Abduh Sayid Albana[2]**

[1]*Computer Engineering, Telkom University, Indonesia*
[2]*Industrial Engineering, Institut Teknologi Telkom Surabaya, Indonesia*
* Corresponding author's Email: purbodaru@telkomuniversity.ac.id

Abstract: Intermediate storage is one important aspect of parallel permutation flow-shop scheduling (PPFSP). Unfortunately, research about optimizing intermediate storage is rare. Most existing studies were conducted to minimize or optimize the maximum completion time under various circumstances and constraints. Thus, this work aims to minimize intermediate storage with a tolerable maximum completion time level. In this work, the proposed model was developed by using an evolutionary algorithm (EA). This model consists of two steps. In the first step, the EA is used to determine the tolerable maximum completion time. In the second step, the EA is used to make a solution with minimum intermediate storage and a tolerable maximum completion time. This proposed model is then compared with the existing non-dominated sorting genetic algorithm (NSGA II), improved efficient genetic algorithm (IEGA), cloud-theory based simulated annealing (CTSA), and artificial bee colony algorithm (ABC). Based on the simulation results, the proposed model performs better than the IEGA, CTSA, and ABC models. Meanwhile, the proposed model is competitive enough compared with the NSGA II. And, the two-step evolutionary algorithm (TSEA) produces 26 percent higher maximum intermediate storage, and the improved two-step evolutionary algorithm (ITSEA) produces 7 percent higher maximum intermediate storage than the NSGA II. The ITSEA produces 16 lower maximum intermediate storage than the TSEA. Both TSEA and ITSEA produce a 21 percent higher maximum completion time than the NSGA II.

Keywords: Parallel flow-shop scheduling, Evolutionary algorithm, Intermediate storage.

1. Introduction

It is widely known in manufacturing that there are several types of production, such as: job-shop, batch, and flow shop. Flow-shop is commonly used in factories with a mass production process. The problem of scheduling a flow-shop, commonly named the flow-shop scheduling problem (FSP), isn't trivial and has been studied widely.

Flow-shop scheduling is a type of job-shop scheduling [1]. It is a processing system in which each job's task sequence is fully specified and all jobs are processed in the same order at the workstations [2]. Flow-shop scheduling determines the best order for jobs to be processed on machines in the same order, i.e. each job must be handled in the same order on machines 1, 2, ..., m [1].

One of FSP's attractiveness is the complexity of the production process. Thus, FSP has many derivatives, such as classical model, permutation, non-permutation, parallel, two-machine, hybrid, etc. Research on FSP was conducted with specific constraints and the processes ran in specific circumstances, such as deteriorating jobs [3], sequence-dependent setup times [4], multiple requirements from customers [5], limited waiting times [4], no-idle machines [6], and blocking limitations [7].

FSP, as an optimization model, is designed to solve problems based on its objective. Several studies applied a single objective, while other studies conducted multi-objective solutions. The majority of FSP research intended to minimize the maximum completion time [8]. Other objectives include tardiness [9], penalties [5], energy consumption [10], and so on.

Ironically, intermediate storage as an objective was rarely investigated, even though it is a crucial part of the flow-shop system. In most FSP research, intermediate storage was considered as a limitation rather than an objective. Several studies assumed that intermediate storage is limitless, whereas others considered that it is limited [11] or nil [7], resulting in the possibility of blocking [12].

This paper presents a parallel FSP model which focuses on minimizing intermediate storage size while maintaining a tolerable maximum completion time. This model can be called a semi-multi-objective model. The maximum completion time can be viewed as an objective as well as a constrain. As a parallel FSP, there are a set number of identical production lines [13], and any job can be processed on any line of the system. As a permutation FSP, a task cannot overlap with other jobs in a sequence once it has been arranged [11].

The evolutionary algorithm [14] is a well-known metaheuristic algorithm that has been widely used to tackle combinatorial problems. It has proven to be a generic, robust and powerful search mechanism [15]. Thus, this model is developed based on an evolutionary algorithm. Other metaheuristic algorithms that have been widely used in solving the FSP are genetic algorithm [3], simulated annealing [16], tabu search [17], particle swarm optimization [9], chaotic-student-psychology based optimization [18], etc.

In this paper, a novel two-step EA solution is proposed. This proposed model consists of two serial steps. In the first step, the algorithm focuses on determining the tolerable maximum completion time level. In the second step, the algorithm focuses on finding the solution that its intermediate storage is minimum.

The contribution of this work is as follows:

(1) This work promotes a novel serial evolutionary-algorithm-based model to solve the parallel permutation flow-shop scheduling problem.

(2) This work proposes a semi-multi-objective problem by concerning the intermediate storage with a tolerable maximum completion time, rather than a pure multi-objective problem where the intermediate storage and maximum completion time are concerned at the same time.

The rest of the paper is organized as follows. The second section discusses the literature on permutation FSP, particularly parallel permutation FSP. In the third section, the proposed two-step EA model is explained. The fourth section explains the simulation results that compare the proposed model's performance to that of existing models. The fifth section delves deeper into the proposed model's analysis, findings, and limits. The conclusion relates to the main result, findings, and the research purpose; and the implications and future research potential are explained in the sixth section.

2. Literature review

The literature on FSP is discussed in this section. Based on [3], the basic process in a flow-shop system is as follows. There are a certain number of jobs. A production process is divided into several stages serially. In a classic FSP, there is a single production line that is divided into stages. There is a machine in each stage. A job must pass through all stages completely. A job enters a stage once [19]. On the other hand, a stage (machine) can only do a job at one time. A job can move to the next stage if it meets two conditions. This job has finished the current stage and the machine in the next stage is available. When this job has finished its current stage but the machine in the next stage has not been available yet, this job must be stored in the intermediate storage. Otherwise, this job is stuck in this current job and blocks other jobs from occupying the machine in the current stage [12].

This classic FSP is divided into two derivatives: the permutation FSP and the non-permutation FSP. In the permutation FSP, once a job sequence is arranged, this sequence does not change until all jobs in the sequence are processed completely [20]. This model applies to the first-in-first-out (FIFO) mechanism [13]. It means jobs overlapping are not permitted. This process is simple. Unfortunately, this permutation FSP may cause longer total completion time. Meanwhile, in the non-permutation FSP, job overlapping is permitted [11]. The goal of non-permutation FSP is to reduce total completion time. A job with a shorter processing time can be processed earlier than other jobs in front of it in the sequence.

There are plenty of studies in the FSP. Several of the latest classic FSPs are as follows. Xuan, Zhang, and Li [3] proposed model for a flexible FSP. Rather than a fixed processing time, in the real world, the processing time may be longer due to several problems: failure, machine, or workers [3]. This problem is known as deteriorating jobs. They formulated this problem by using mixed integer linear programming (MILP). This work aimed to minimize total weighted completion time. They used a combined artificial bee colony algorithm and a genetic algorithm to solve this problem.

Hsu, Lin, Duan, Liao, Wu, and Chen [16] proposed incorporating scenario-dependent processing times into a two-machine flow-shop environment in order to reduce total completion time. They first derive a lower bound and two optimality properties to improve the branch-and-bound method's searching efficiency. Then, using a pairwise interchange mechanism, they offer 12 basic heuristics and their corresponding counterparts. Furthermore, they presented 12 simple heuristics as the 12 starting seeds for designing 12 cloud theory-based simulated annealing (CSA) variants.

Lee [4] proposed a model for two-machine FSP. The first constraint is waiting time. The second constraint is setup time. This work aimed to minimize tardiness. The problem was formulated by using MILP. This work used genetic algorithms to solve the problem. Several assumptions in this work were as follows. The information about the processing time, sequence-dependent setup time, limited waiting time, and due date are known in advance.

Assia, El-Abbassi, El-Barkany, Darcherif, and El-Biyaali. [10] proposed a green scheduling model for two-machine permutation FSP. This work integrated energy consumption and maintenance scheduling as constraints. This work aimed to minimize total completion time and energy consumption. This problem was formulated by using MILP. A machine can only process a job at one time, and a job can only be processed by one machine at one time.

The next derivative in FSP is parallel FSP. It is a system that consists of several parallel identical processing units or identical lines [9]. Jobs are distributed into production lines. A parallel FSP can be permutation or non-permutation. In its classic mode, once a job is allocated to a certain line, it will stay on this line until its process finishes completely. It cannot jump to the other line during the processing time, although the machine in the next stage is available or it can be produced better (faster) in the other line for the next stage. Fig. 1 illustrates the parallel FSP. There is plenty of research in parallel with FSP.

Here are a few of the most recent ones. Sun and Qi [5] conducted research in the parallel FSP where there are multiple requirements for every job. This was about a customized job due to a customer's request. Based on this customization, stages that must be passed by every job might be different from each other. Jobs of the same type would pass through the same number of stages. This work aimed to minimize the penalty due to lateness and the maximum completion time. The authors proposed a combination of ant colony optimization and simulated annealing algorithms to solve the problem. The proposed model combines differential evolution and local search algorithms. In this work, interruptions were not allowed, and the arrival time of all jobs was known in advance.

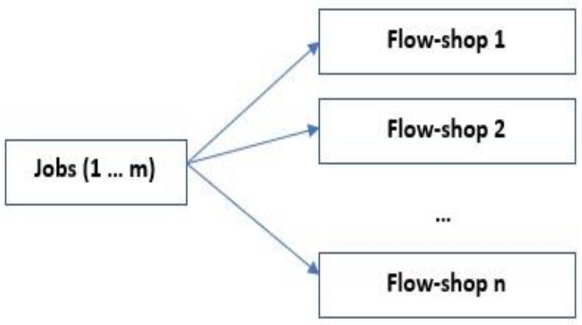


Figure. 1 Parallel FSP illustration

Geng, Ye, Cao, and Liu. [19] conducted parallel FSP research where re-entrance mechanism was possible. Rather than in classic FSP, where a job enters a stage once, a job might enter the same stage more than once. This condition occurs in certain industries. Energy consumption has become an important aspect. This work aimed to minimize maximum completion time, maximum tardiness, and idle energy consumption. The stop-and-go strategy was proposed. A machine will be turned off when it is idle to save energy. This problem was formulated by using MILP. Machine failure and machine adjustment were excluded from the model. They used a multi-verse optimizer algorithm, one kind of heuristic technique, to solve the problem.

Ribas & Companys [7] developed a model for parallel FSP that allowed blocking mechanism. This work aimed to minimize the maximum completion time. The blocking mechanism occurs due to the absence of the intermediate storage that plays as a buffer for the job during the inter-stage condition. Because there was no intermediate storage, a job must wait to proceed to the next stage while the adjacent machine is still working. Several heuristic methods were compared: Palmer heuristic (PAL), shortest processing time (SPT), largest processing time (LPT), and trapezium (TRA).

The other type of parallel FSP is the distributed FSP. There are a certain number of jobs that are distributed to a certain number of factories. Each factory is identical, in the context that it has the same set of the same number of machines [26]. Several studies on distributed FSP are as follows. Fernandez-Viagas & Framinan [27] proposed the bounded-search-integrated greedy algorithm to solve

Received: August 13, 2021. Revised: September 13, 2021.

Table 1. Previous works in flow-shop scheduling

Work	Objective Parameters	Method
[8]	make-span	MILP and 12 heuristic methods
[21]	make-span	scatter search (SS)
[19]	make-span, tardiness, and idle energy consumption.	multi-objective multi-verse optimizer (IMOMVO)
[22]	make-span	iterated greedy algorithm
[3]	total weighted completion time	artificial bee colony algorithm
[10]	make-span, a measure of service level and total energy consumption	MILP
[16]	total completion time.	cloud theory-based simulated annealing (CSA)
[23]	total actual flow time	MILP
[24]	total weighted tardiness	constructive heuristics and branch-and-bound algorithm (B&B)
[4]	total tardiness	GA
[5]	make-span and the cost of delay	differential evolution (DE) and local search (LS)
[12]	make-span	MILP and iterated greedy algorithm
[25]	total completion time	IEGA
[9]	total weighted tardiness, total operation time, and total cost of the company's reputational damage	MILP, MOPSO, NSGA II
[7]	maximum completion time of jobs	constructive heuristics

the distributed permutation FSP. Naderi & Ruiz [8] compared 14 heuristic-based dispatching methods to solve problems in distributed permutation FSP. Their work aimed to minimize the make-span. Overall, the summary of the latest studies in FSP is shown in Table 1.

Table 1 is organized based on chronological order. It can be seen that parallel FSP is still an interesting issue to be discussed. Based on our literature review, we decided to explore more about parallel permutation flow-shop scheduling (PPFSP). Similar to the existing studies, we minimize or optimize the maximum completion time or make-span with various circumstances and constraints. However, we chose to specialize in optimizing the intermediates, as research in this field is rare. Most of the existing research uses the genetic algorithm variant where GA is one of evolutionary computations. We chose another type of evolutionary computation, named the evolutionary algorithm (EA). Our proposed model will be compared with these existing ones.

3. Model

As a parallel permutation FSP, the system consists of several shops. Each shop consists of a production line with a certain number of stages. In every stage, there is a machine or set of machines. There is a set of jobs that enter the system to be produced in any shop. This model is developed based on several assumptions as followed:

- The jobs are homogeneous. It means the number of stages that must be passed by every job is same [5].
- Every job can be produced by any shop in the system [8].
- Pre-emption is not allowed [6].
- The jobs are independent [19].
- As a parallel FSP, once a sequence is determined, a job cannot overlap with other jobs in a sequence during the production process [21].
- All jobs are ready at time zero [4].
- All machines are ready at time zero.
- The processing time of every job in every stage in every machine is known in advance [4].
- The setup time is zero. It means that once a job finishes in a stage and the machine in the next stage is available, this job can move to the next stage immediately.
- Every shop has intermediate storage so that once a job finishes in a stage and the machine in the next stage is still unavailable, this job is transferred to the intermediate storage so that this job does not blocks the next job in the sequence.
- The intermediate storage has unlimited capacity.

We use several notations in order to model our problems. The notations that are used in this work are as followed:

f_{th}	:	threshold factor
g	:	stage
G	:	set of stages in a processing line
j	:	job
J	:	set of jobs
j_{sm}	:	selected job for mutation
l	:	solution
L	:	set of solutions
n	:	number of a set

Received: August 13, 2021. Revised: September 13, 2021.

n_v	:	intermediate storage
o	:	offspring
p_s	:	shop of the job
p_e	:	position of a job in a sequence
q	:	sequence
Q	:	set of sequence in a shop
s	:	shop
S	:	set of shops
s_{sm}	:	selected shop for mutation
t_a	:	start time of a job in a stage
t_{acl}	:	the all-time lowest maximum completion time during iteration
t_{ach}	:	the all-time highest make-span during iteration
t_c	:	maximum completion time of a shop
t_{cl}	:	the lowest maximum completion time of a solution
t_{ch}	:	the highest maximum completion time of a solution
t_e	:	end time of a job in a stage
t_i	:	iteration time
T_i	:	set of iteration time
t_p	:	processing time
T_p	:	set of processing time
t_{trc}	:	the maximum completion time threshold
t_{th}	:	threshold of the tolerable maximum completion time
v	:	the maximum intermediate storage in a solution
v_{al}	:	the all-time lowest maximum intermediate storage
v_h	:	the highest maximum intermediate storage among solutions
v_l	:	the lowest maximum intermediate storage among solutions

In this work, the proposed model to solve the parallel permutation FSP is developed based on the evolutionary algorithm (EA). EA is a popular metaheuristic algorithm. This condition is similar to the parallel permutation FSP, which is also a combinatorial problem. EA is a population-based solution that is developed based on the evolution system [28]. The better solution is found by creating new generations, mutations, and crossovers.

During iterations, new generations are generated by the fittest or the best solution from the current generation, and the worst current solution is replaced by this offspring [29]. The illustration of basic EA is shown in Fig. 2. The basic algorithm of the EA is as follows [28].

1. In the beginning, certain number of solutions are generated randomly. The number of solutions represents the population size.

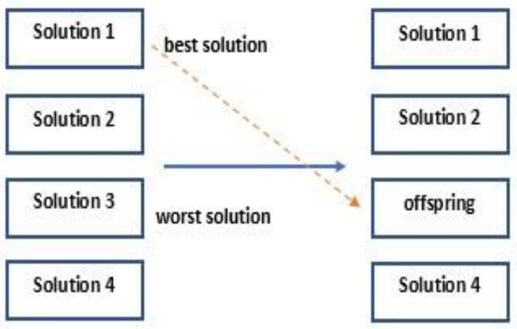


Figure. 2 Evolutionary algorithm illustration

2. The fitness value of every solution is calculated by using a pre-determined fitness function.
3. Based on this fitness function, the best solution and the worst solution are determined.
4. The worst solution is replaced by the best solution as the offspring.
5. Mutation is conducted to the offspring so that the offspring becomes new solution.
6. Step 2 until step 5 is repeated in every iteration.
7. The iteration stops if one of these two conditions occurs. First, the maximum iteration is reached. Second, the targeted fitness value is reached.
8. The quantity of all jobs is homogeneous.

A solution represents a parallel permutation FSP that consists of a number of shops and a number of jobs in every shop. This solution is represented as a two-dimensional array. The first index of the array represents the shop. The second index of the array represents the job order in a shop. Each cell represents a job. This array is illustrated in Fig. 4. For example, in Fig. 3, a population consists of 12 jobs and 3 shops. There are 4 jobs in every shop.

In this work, the mutation process occurs by exchanging jobs in a selected shop with another job in another selected shop. This mutation occurs in a solution. This mutation process is formalized by using Eqs. (1) and (2). This mutation is illustrated in Fig. 4. Eq. (1) shows that the selected shop that its job will mutate is chosen at random and follows a

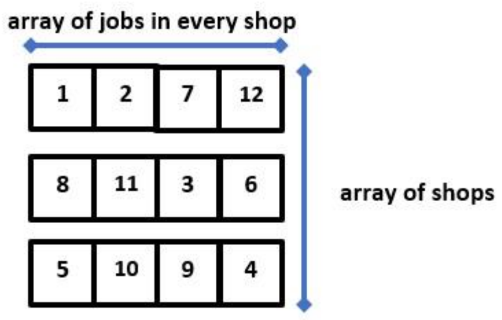


Figure. 3 Array design of a solution

Received: August 13, 2021. Revised: September 13, 2021.

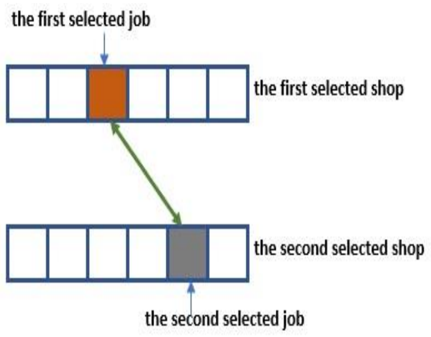


Figure. 4 Mutation process

uniform distribution. Eq. (2) shows that the position of a job in a selected shop that mutates is randomly determined and follows a uniform distribution.

$$s_{sm} = U\big(1, n(S)\big) \tag{1}$$

$$p(j_{sm}) = U\big(1, n(J_{sh})\big) \tag{2}$$

In this work, two models are proposed: Two-Step Evolutionary Algorithm (TSEA) and Improved Two-Step Evolutionary Algorithm (ITSEA). Between these two models, the difference lies in the number of parents that will produce offspring in every iteration. In the TSEA, only one solution becomes parent. In the ITSEA, half of the solutions become parents. This idea is inspired by the NSGA II where the parents are half of the sorted population [30]. In every model, the proposed model consists of two serial EA steps. The difference between TSEA and ITSEA occurs only in the second step.

In the first step, the TSEA process is conducted to find the lowest and the highest maximum completion time. These variables are used to determine the maximum completion time threshold that will be used in the second step. The algorithm for the first step is as follows.

1. In the beginning, certain number of solutions are generated randomly.
2. The maximum completion time of every solution is calculated.
3. The lowest and the highest maximum completion time is calculated.
4. The all-time lowest and all-time highest maximum completion time is updated.
5. The offspring is generated by replacing the solution with the highest maximum completion time with the solution with the lowest maximum completion time.
6. This offspring mutates.
7. Steps 2 to 6 are repeated.
8. Iteration stops when the maximum iteration is reached.

The algorithm process is formalized by using Eqs. (3) to (13).

$$p_s(j) = U\big(1, n(S)\big), t = 0 \tag{3}$$

$$p_q(j) = U\big(1, n(Q)\big), t = 0 \tag{4}$$

$$t_{sc}(s, t_i) = t_e(j, g), p_q(j) = n(Q) \wedge p_g = n(G) \tag{5}$$

$$t_c(l, t_i) = max\big(t_{sc}(s, t_i)\big), s \in S_l \tag{6}$$

$$t_{cl}(t_i) = min\big(t_c(l, t_i)\big), l \in L \tag{7}$$

$$t_{ch}(t_i) = max\big(t_c(l, t_i)\big), l \in L \tag{8}$$

$$t_{acl} = min\big(t_{cl}(t_i)\big), t_i \in T_i \tag{9}$$

$$t_{ach} = max(t_{ch}(t_i)), t_i \in T_i \tag{10}$$

$$o(t_i) = l, t_c(l, t_i) = t_{cl}(t_i) \tag{11}$$

$$p_l\big(o(t_i)\big) = p_l(l), t_c(l, t_i) = t_{ch}(t_i) \tag{12}$$

$$t_{th} = t_{acl} + \big(f_{th}.(t_{ach} - t_{acl})\big) \tag{13}$$

Eq. (3) shows that in the beginning, a job is distributed randomly among shops in a solution. It follows a uniform distribution. Eq. (4) shows that in the beginning, the position of a job in the sequence is randomly distributed. It follows a uniform distribution. Eq. (5) shows that the completion time in a shop is the last stage end time of the last job in the shop. Eq. (6) shows that the maximum completion time of a solution is the highest completion time among shops in this solution. Eq. (7) shows that the lowest maximum completion time is the lowest maximum completion time among solutions in an iteration. Eq. (8) shows that the highest maximum completion time is the highest maximum completion time among solutions in an iteration. Eq. (9) shows that the all-time lowest maximum completion time is the lowest maximum completion time during the iteration. Eq. (10) shows that the all-time highest completion time is the highest completion time during the iteration. Eq. (11) shows that the offspring is the solution with the lowest maximum completion time. Eq. (12) shows that the offspring will replace the solution with the highest maximum completion time. Eq. (13) shows that the maximum completion time threshold is determined by the range between the all-time lowest maximum completion time and the all-time highest maximum completion time, and it is multiplied by the threshold factor.

In the second step, the TSEA process is conducted to find the lowest intermediate storage with the tolerable maximum completion time. The algorithm for the second step is as follows.

1. In the beginning, certain number of solutions are generated randomly.
2. The maximum completion time of every solution is calculated. The lowest and the highest intermediate storage is calculated.
3. If the solution that its maximum intermediate storage is the lowest among other solutions in this iteration and its maximum completion time is equal or below the maximum completion time threshold, then the all-time lowest maximum intermediate storage is updated.
4. The lowest and the highest maximum intermediate storage is calculated.
5. The offspring is generated by replacing the solution with the highest maximum intermediate storage with the solution with the lowest maximum completion time.
6. This offspring mutates.
7. Steps 2 to 6 are repeated.
8. Iteration stops when the maximum iteration is reached.

The algorithm process is formalized by using Eqs. (14) to (21).

$$n_v(s,t_p,t_i) = \sum_{\forall j, p_s(j)=s} j, \left(t_e(j,g) < t_p < t_a(j,g+1) \wedge g < n(G)\right) \tag{14}$$

$$v(l,t_i) = max\left(n_v(s,t_p,t_i)\right), s \in S_l \wedge t_p \in T_p(t_i) \tag{15}$$

$$v_h(t_i) = max\left(v(l,t_i)\right), l \in L \tag{16}$$

$$v_l(t_i) = min\left(v(l,t_i)\right), l \in L \tag{17}$$

$$o(t_i) = l, v(l,t_i) = v_l(t_i) \tag{18}$$

$$p_l\left(o(t_i)\right) = p_l(l), v(l,t_i) = v_h(t_i) \tag{19}$$

$$v_{al}(t_i) = v_l(t_i), t_i = 1 \tag{20}$$

$$v_{al}(t_i) = \begin{cases} v_{al}(t_i-1), v_{al}(t_i-1) \le v_l(t_i) \\ v_l(t_i), v_{al}(t_i-1) > v_l(t_i) \end{cases}, t_i > 1 \tag{21}$$

Eq. (14) shows that the intermediate storage is detected by calculating the number of jobs which have idle time at a certain processing time. Eq. (15) shows that the maximum intermediate storage of a solution is calculated by finding the maximum number of shops that need intermediate storage at a certain processing time along the processing time. Eq. (16) shows that the highest maximum intermediate storage is the highest maximum intermediate storage among the solutions in an iteration time. Eq. (17) shows that the lowest maximum intermediate storage is the lowest maximum intermediate storage among the solutions in an iteration time. Eq. (18) shows that the offspring is generated from the solution with the lowest maximum intermediate storage. Eq. (19) shows that the offspring replaces the solution with the highest maximum intermediate storage. Eq. (20) shows that in the beginning, the all-time lowest maximum intermediate storage is the lowest maximum intermediate storage. Eq. (21) shows that the all-time lowest maximum intermediate storage is updated when the current lowest maximum intermediate storage is lower than the previous lowest maximum intermediate storage for iteration time is more than 1.

In the second step of the ITSEA, as mentioned above, in every iteration, the number of parents is half of the population. The algorithm of the ITSEA is as follows.

1. In the beginning, certain number of solutions are generated randomly.
2. The maximum completion time of every solution is calculated.
3. The solutions are then sorted ascendingly based on the maximum intermediate storage.
4. If the solution that its maximum intermediate storage is the lowest among other solutions in this iteration and its maximum completion time is equal or below the maximum completion time threshold, then the all-time lowest maximum intermediate storage is updated.
5. Half best of the population is selected as parents.
6. These selected parents then become offspring after mutated.
7. These offspring then replace the half worst of the population.
8. Steps 2 to 6 are repeated.
9. Iteration stops when the maximum iteration is reached.

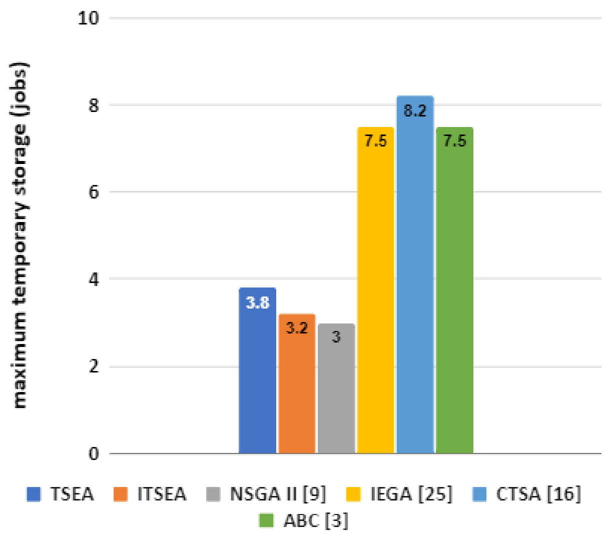
To see performance of our proposed model, we are going to compare our model with the four techniques: NSGA II [9], IEGA [25], CTSA [16], and ABC [3]. These four comparing algorithms were used in several latest FSP studies as they are explained in the literature review.

4. Simulation and result

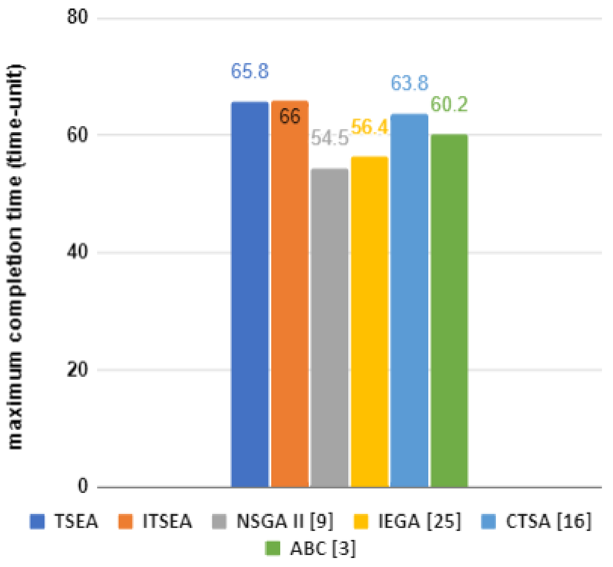
This model is then implemented into simulation to evaluate its performance. In this simulation, the proposed model is then compared with the existing NSGA II [9], IEGA [25], CTSA [16], and ABC [3]. In this simulation, the objectives of the NSGA II are modified so that the objectives are maximum completion time and maximum intermediate storage. The goal is to minimize these two parameters. Meanwhile, the IEGA [25], CTSA [16], and ABC [3] is a single-objective model whose objective is to minimize the maximum completion time. The CTSA [16] and ABC [3] are chosen due to their feature in avoiding local optimal trap. In this work, two simulations were conducted.

In the first simulation, the adjusted variables are similar to NSGA II [9]. There are 40 jobs, 5 shops, and 4 stages. The average processing time in every stage is 5 time-units. It is randomly generated and follows normal distribution. The population size is 20. In NSGA II, the parent population size is 10 because the total population is doubled [30]. In every simulation session, there are 200 iterations. The maximum completion time threshold is 0.5. In this simulation, there are two observed parameters: maximum intermediate storage and maximum completion time. The simulation result is shown in Fig. 5.

In Fig. 5a, the NSGA II [9] becomes the best solution for producing the lowest maximum intermediate storage. Meanwhile, the CTSA [16] becomes the worst solution. Although the proposed models produce higher maximum intermediate storage rather than the NSGA II, their value is competitive enough compared with the IEGA [25], CTSA [16], and ABC [3]. The IEGA and ABC produces higher maximum temporary storage rather than the proposed model, but still lower than the CTSA. By comparing the proposed models, the ITSEA produces lower maximum intermediate storage than the TSEA. The TSEA produces 26 percent higher maximum intermediate storage than the NSGA II. The ITSEA produces 7 percent higher maximum intermediate storage than the NSGA II. The TSEA produces 57 percent lower maximum intermediate storage than the CTSA [16]. Meanwhile, the ITSEA produces 61 percent lower maximum intermediate storage than the CTSA [16].
In Fig. 5b, the NSGA II becomes the best solution for producing the lowest maximum completion time. Meanwhile, ITSEA becomes the worst solution.



(a)



(b)

Figure. 5 Model performance comparison: (a) maximum temporary storage and (b) maximum completion time

The TSEA produces a 21 percent, 17 percent, 3 percent, and 9 percent higher maximum completion time than the NSGA II [19], IEGA [25], CTSA [16], and ABC [3] consecutively.

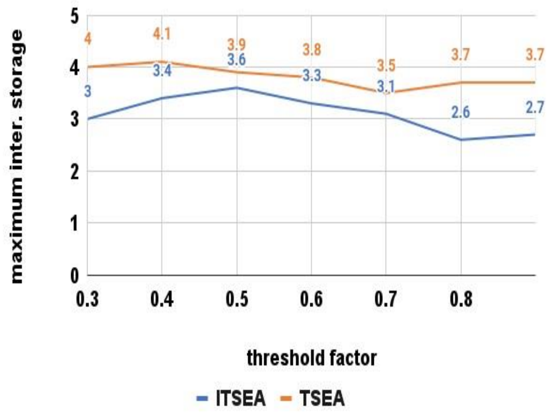
In the second simulation, the relationship between the threshold factor and the observed parameters is conducted. The reason why this simulation was conducted is because in the proposed model, the system may not be able to produce a solution because after the maximum iteration is reached, the system still cannot find a solution where the maximum completion time is under the threshold. The observed factors include maximum intermediate storage, maximum completion time, and success percentage. The success percentage is the percentage between the number of simulations that produce solutions and the total number of solutions. The adjusted parameters, which are jobs, shops, stages, processing times, and population size, are the same as in the first simulation. The evaluated models in this simulation are TSEA and ITSEA. The result is shown in Fig. 6.

Fig. 6 shows that there is a relationship between the threshold factor and the observed parameters. Fig. 6a shows that the increase in the threshold factor makes the maximum intermediate storage decrease. It occurs in both TSEA and ITSEA. In all threshold factor values, the ITSEA produces lower maximum intermediate storage than the TSEA. Fig. 6b shows that the increase in the threshold factor makes the maximum completion time increase. It occurs in both TSEA and ITSEA. The maximum completion time of both models is almost equal in all threshold factor values. Fig. 6c shows that the increase in the threshold factor makes the success percentage increase when the threshold factor is from 0.3 to 0.7. And after that, the success percentage is stagnant because the success percentage is almost or equal to 100 percent. It occurs in both models. The TSEA and ITSEA produce similar value and trend in success percentage in all threshold factor values.

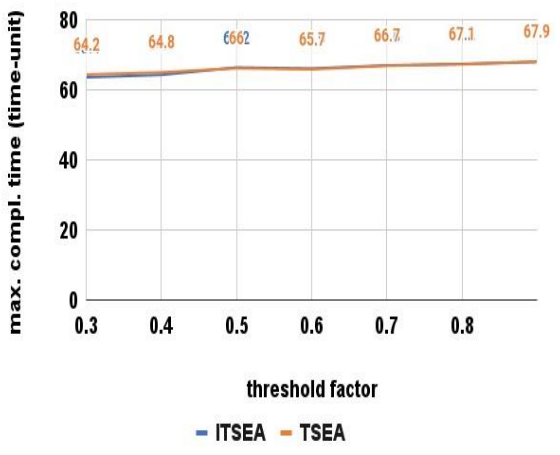
5. Discussion

There are several findings in this work. The proposed models, both TSEA and ITSEA, are able to achieve the objective of this work, which is to minimize the maximum intermediate storage with a tolerable maximum completion time. Both models are better than the single objective models, the IEGA, CTSA, and ABC models, whose objective is to minimize maximum completion time. These proposed models are competitive enough compared with the multi-objective solution. Meanwhile, the proposed models are much better than the single objective models (IEGA [25], CTSA [16], and ABC [3]), which represents the majority of existing studies in flow-shop scheduling which most of their objectives are to minimize the maximum completion time.

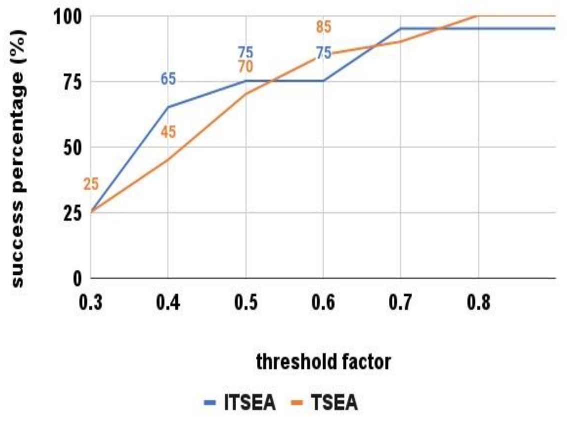
Unfortunately, although the proposed model can produce solutions at a tolerable maximum completion time, its maximum completion time is still higher than all comparing previous models (IEGA, NSGA II, CTSA, and ABC). Among these previous models, the NSGA II becomes the best solution in both parameters due to its competitiveness in producing pareto-optimal solutions [30]. This result, where the NSGA II creates the best in both (all) parameters, while the proposed models can compete in minimizing



(a)



(b)



(c)

Figure. 6 Relation between threshold factor and the observed parameters: (a) maximum temporary storage, (b) maximum completion time, and (c) success percentage

maximum intermediate storage but give up on maximum completion time, also shows the concept difference between both models. In NSGA II, as a multi-objective solution, it tries to minimize all parameters by finding the most dominant solution [30]. On the other hand, the proposed models, as semi-multi-objective solutions, try to minimize maximum intermediate storage while trying to keep the maximum completion time low within the tolerable range. This result also proves that the IEGA, CTSA, and ABC, as a single-objective solution, focus on its single objective while ignoring other parameters [9].

Our proposed model has a weakness that there is a probability where the model fails to find a solution with a tolerable maximum completion time threshold. This probability is high when the threshold factor is low. It is because when the tolerable maximum completion time is low, finding an accepted solution becomes more difficult, although its maximum completion time is better. On the other hand, finding an accepted solution becomes easier when the threshold is higher, although it makes the production maximum completion time increase.

The ITSEA performs better than the TSEA, especially in producing low maximum intermediate storage. The difference between the TSEA and ITSEA is in the number of offspring that are produced in every iteration. Based on this, it can be said that more offspring makes the performance of this EA based model better at achieving its objective. On the other hand, the number of offspring does not affect the tolerable maximum completion time, which is in the second step, it becomes a constraint.

6. Conclusion

This work has demonstrated that the proposed models, two-step evolutionary algorithm (TSEA) and improved two-step evolutionary algorithm (ITSEA), can solve the problem of minimizing the maximum intermediate storage with the maximum completion time threshold in the parallel flow-shop system. Both models are competitive enough with the existing NSGA II, although the NSGA II is a little bit better, and much better than the existing single objective improved efficient genetic algorithm (IEGA), cloud-theory based algorithm (CTSA), and artificial bee colony algorithm (ABC). Overall, the ITSEA is better than the TSEA. Unfortunately, the maximum completion time produced by the proposed models, although still tolerable, is worse than the existing comparing methods. Based on the simulation results, the TSEA produces 26 percent higher maximum intermediate storage and the ITSEA produces 7 percent higher maximum intermediate storage than the NSGA II. The ITSEA produces 16 lower maximum intermediate storage than the TSEA. Both TSEA and ITSEA produce a 21 percent higher maximum completion time than the NSGA II. Meanwhile, there is a weakness in these proposed models. The proposed models may fail to find an accepted solution during the iteration. This failure probability is high when the threshold factor is low or when the tolerable maximum completion time range is tight. On the other hand, this failure probability is low when the threshold factor is high. Unfortunately, the increase in the threshold factor makes the maximum completion time increase.

This model has the potential to be improved in the future. The first improvement could be to improve its competitiveness compared with multi-objective solutions, such as NSGA II. The second improvement could be to reduce the failure probability of finding an accepted solution, especially in the tight tolerable attributes range.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization: Kusuma; methodology: Kusuma and Albana; software: Kusuma; validation: Kusuma; formal analysis: Kusuma and Albana; investigation: Kusuma and Albana; writing-original draft preparation: Kusuma; writing-review and editing: Albana.

Acknowledgments

This work is funded by Telkom University.

References

[1] S. Kumar and P. Jadon, "A Novel Hybrid Algorithm for Permutation Flow Shop Scheduling", *International Journal of Computer Science and Information Technologies*, Vol. 5, No. 4, pp. 5057-5061, 2014.

[2] H. Emmons and G. Vairaktarakis, *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*, Springer Science & Business Media, 2012.

[3] H. Xuan, H. Zhang, and B. Li, "An Improved Discrete Artificial Bee Colony Algorithm for Flexible Flowshop Scheduling with Step Deteriorating Jobs and Sequence-dependent Setup Times", *Mathematical Problems in Engineering*, pp. 1-13, 2019.

[4] J. Y. Lee, "A Genetic Algorithm for a Two-machine Flowshop with a Limited Waiting Time Constraint and Sequence-dependent

Setup Times", *Mathematical Problems in Engineering*, pp. 1-13, 2020.

[5] Y. Sun and X. Qi, "A DE-LS Metaheuristic Algorithm for Hybrid Flow-shop Scheduling Problem Considering Multiple Requirements of Customers", *Scientific Programming*, pp. 1-14, 2020.

[6] D. Gupta and H. Singh, "A Heuristic Approach to n x m Flow Shop Scheduling Problem in which Processing Times are Associated with Their Respective Probabilities with No-idle Constraint", *ISRN Operations Research*, pp. 1-9, 2013.

[7] I. Ribas and R. Companys, "A Computational Evaluation of Constructive Heuristics for the Parallel Blocking Flow Shop Problem with Sequence-dependent Setup Times", *International Journal of Industrial Engineering Computations*, Vol. 12, pp. 321-328, 2021.

[8] B. Naderi and R. Ruiz, "The Distributed Permutation Flowshop Scheduling Problem", *Computers & Operations Research*, Vol. 37, pp. 754-768, 2010.

[9] N. Farmand, H. Zarei, and M. R. Barzoki, "Two Meta-heuristic Algorithms for Optimizing a Multi-objective Supply Chain Scheduling Problem in an Identical Parallel Machines Environment", *International Journal of Industrial Engineering Computations*, Vol. 12, pp. 249-272, 2021.

[10] S. Assia, I. E. Abbassi, A. E. Barkany, M. Darcherif, and A. E. Biyaali, "Green Scheduling of Jobs and Flexible Periods of Maintenance in a Two-machine Flowshop to Minimize Makespan, a Measure of Service Level and Total Energy Consumption", *Advances in Operations Research*, pp. 1-9, 2020.

[11] D. Rossit, F. Tohme, M. Frutos, M. Safe, and O. C. Vasquez, "Critical Paths of Non-permutation and Permutation Flow Shop Scheduling Problems", *International Journal of Industrial Engineering Computations*, Vol. 11, pp. 281-298, 2020.

[12] M. Takano and M. Nagano, "Solving the Permutation Flow Shop Problem with Blocking and Setup Time Constraints", *International Journal of Industrial Engineering Computations*, Vol. 11, pp. 469-480, 2020.

[13] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, 2018.

[14] A. P. Engelbrecht, *Computational Intelligence: an Introduction*, John Wiley & Sons, 2007.

[15] S. Bhargava, "A Note on Evolutionary Algorithms and Its Applications", *Adults Learning Mathematics*, Vol. 8, pp. 31-45, 2013.

[16] C. L. Hsu, W. C. Lin, L. Duan, J. R. Liao, C. C. Wu, and J. H. Chen, "A Robust Two-machine Flow-shop Scheduling Model with Scenario-dependent Processing Times", *Discrete Dynamics in Nature and Society*, pp. 1-16, 2020.

[17] F. Cetinkaya, P. Yeloglu, and H. Catmakas, "Customer Order Scheduling with Job-based Processing on a Single-machine to Minimize the Total Completion Time", *International Journal of Industrial Engineering Computations*, Vol. 12, pp. 273-292, 2021.

[18] A. Sasmito and A. B. Pratiwi, "Chaotic Student Psychology based Optimization Algorithm for Bi-objective Permutation Flowshop Scheduling Problem", *International Journal of Intelligent Engineering and System*, Vol. 14, No. 3, pp. 109-118, 2021.

[19] K. Geng, C. Ye, L. Cao, and L. Liu, "Multi-Objective Reentrant Hybrid Flowshop Scheduling with Machines Turning On and Off Control Strategy Using Improved Multi-verse Optimizer Algorithm", *Mathematical Problems in Engineering*, pp. 1-18, 2019.

[20] A. Mishra and D. Shrivastava, "A Discrete Jaya Algorithm for Permutation Flow-shop Scheduling Problem", *International Journal of Industrial Engineering Computations*, Vol. 11, pp. 415-428, 2020.

[21] B. Naderi and R. Ruiz, "A Scatter Search Algorithm for the Distributed Permutation Flowshop Scheduling Problem", *European Journal of Operational Research*, Vol. 239, pp. 323-334, 2014.

[22] R. Ruiz, Q. K. Pan, and B. Naderi, "Iterated Greedy Methods for the Distributed Permutation Flowshop Scheduling Problem", *Omega*, Vol. 83, pp. 213-222, 2019.

[23] D. Kurniawan, A. C. Raja, S. Suprayogi, and A. H. Halim, "A Flow Shop Batch Scheduling and Operator Assignment Model with Time-changing Effects of Learning and Forgetting to Minimize Total Actual Flow Time", *Journal of Industrial Engineering and Management*, Vol. 13, pp. 546-564, 2020.

[24] I. S. Lee, "A Scheduling Problem to Minimize Total Weighted Tardiness in the Two-stage Assembly Flowshop", *Mathematical Problems in Engineering*, pp. 1-10 2020.

[25] M. A. Basset, R. Mohamed, M. Abouhawwash, R. K. Chakrabortty, and M. J. Ryan, "A Simple and Effective Approach for Tackling the

Permutation Flow Shop Scheduling Problem", *Mathematics*, Vol. 9, pp. 1-23, 2021.

[26] Y. Xu, L. Wang, S. Wang, and M. Liu, "An Effective Hybrid Immune Algorithm for Solving the Distributed Permutation Flow-shop Scheduling Problem", *Engineering Optimization*, Vol. 46, pp. 1269-1283, 2014.

[27] V. F. Viagas and J. M. Framinan, "A Bounded-search Iterated Greedy Algorithm for the Distributed Permutation Flowshop Scheduling Problem", *International Journal of Production Research*, Vol. 53, pp. 1111-1123, 2015.

[28] K. Peng, L. Wu, Y. Yi, and X. Chen, "An Effective Hybrid Algorithm for Permutation Flow Shop Scheduling Problem with Setup Time", *Procedia CIRP*, Vol. 72, pp. 1288-1292, 2018.

[29] Suyanto, A. T. Wibowo, S. A. Faraby, S. Saadah, and R. Rismala, "Evolutionary Rao Algorithm", *Journal of Computational Science*, Vol. 53, 2021.

[30] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 6, pp. 182-197, 2002.