An effective hybrid ant lion algorithm to minimize mean tardiness on permutation flow shop scheduling problem

Dana Marsetiya Utama [a,1,*], Dian Setiya Widodo [b,2], Muhammad Faisal Ibrahim [c,3], Shanty Kusuma Dewi [a,4]

[a] University of Muhammadiyah Malang, Malang, Indonesia
[b] University of 17 Agustus 1945 Surabaya, Surabaya, Indonesia
[c] Universitas Internasional Semen Indonesia, Gresik, Indonesia
[1] dana@umm.ac.id; [2] diansetiyawidodo@yahoo.com; [3] faisalibrahim.ie@gmail.com; [4] shanty@umm.ac.id
* corresponding author

ARTICLE INFO

Article history
Received July 23, 2019
Revised December 15, 2019
Accepted February 22, 2020
Available online March 31, 2020

Keywords
Optimization
Mean tardiness
Hybrid Ant Lion
Flow shop
Scheduling

ABSTRACT

This article aimed to develop an improved Ant Lion algorithm. The objective function was to minimize the mean tardiness on the flow shop scheduling problem with a focus on the permutation flow shop problem (PFSP). The Hybrid Ant Lion Optimization Algorithm (HALO) with local strategy was proposed, and from the total search of the agent, the NEH-EDD algorithm was applied. Moreover, the diversity of the nominee schedule was improved through the use of swap mutation, flip, and slide to determine the best solution in each iteration. Finally, the HALO was compared with some algorithms, while some numerical experiments were used to show the performances of the proposed algorithms. It is important to note that comparative analysis has been previously conducted using the nine variations of the PFSSP problem, and the HALO obtained was compared to other algorithms based on numerical experiments.

1. Introduction

Scheduling is the process of delegating tasks to a machine for accomplishment [1], and one of the challenges usually observed is the permutation flow shop problem (PFSP). This is usually observed in the allocation of job sequencing through a machine, M, to achieve a particular performance. In a traditional PFSP, there is mostly idleness of machine during operations through the processing of just one task at a time, and it is directly ready. All the jobs have similar routing on every machine starting from M1 to M2, M3, M4, Mn [2][3], but one problem usually persistent is the minimization of the mean tardiness. Therefore, appropriate scheduling is essential for companies to minimize this problem [4] to have the ability to fulfill orders before the customers' proposed due dates [5] and, consequently, increase customer confidence. Several factors are considered when conducting this process, and the major one is the differences in the penalties associated with the jobs, with the due date being the most prevalent [6].

Several experts have revealed that the PFSP case cannot be found in the polynomial time. Hence, it is included in NP-Hard problems [7]. Therefore, several attempts have been made by researchers to develop an algorithm to reduce tardiness. Some have investigated the problem in a flow shop, for example, Kim [8], Fernandez-Viagas and Framinan [9] developed heuristic algorithms inspired by Nawaz *et al.* [10] while Nagano *et al.* [11] proposed a heuristics algorithm with no idleness. Moreover, a greedy

algorithm was used by Karabulut [6], while Fernandez-Viagas and Framinan [12] developed a non-population-based algorithm. M'Hallah [13] also proposed a hybrid heuristic model, while Kim *et al.* [14] suggested a new priority on flow shop scheduling. Despite the fact there are several kinds of literature on heuristics algorithm, NEH has been reported to perform effectively for PFSP cases [15]. Furthermore, several metaheuristics methods have been proposed such as the Genetic Algorithm (GA) [16], Tabu Search (TS) [17], Particle Swarm Optimizer (PSO) [18], and Simulated Annealing (SA) [19] while some of the new metaheuristics studied include hybrid discrete teaching-learning algorithm [20], Hybrid GA [21], Artificial Immune System, [22] and bi-population EDA [23]. Even though many methods have been proposed to reduce tardiness in PFSP, certain meta-heuristics have drawbacks such as a high computational time when solving large-scale problems and the production of optimal local solutions [24][25].

Recently, the focus of researchers has been on the use of nature algorithms to solve the optimization problem, and several alternative methods have been developed. For this study, the Hybrid Ant Lion Optimization (HALO) algorithm was proposed to solve the PSFP by minimizing the mean tardiness. ALO is an algorithm that mimics the behavior of the Ant Lion and the Ant to survive, and it was first introduced by Mirjalili [26]. It has been used to solve some scheduling problems such as the scheduling in the power system (Chopra and Mehta [27]), Hydro-Thermal Scheduling (Dubey *et al.* [28]), maintenance schedule (Umamaheswari *et al.* [29]) and minimizing makespan in the PFSP (Petrović *et al.* [30]). This means one way to investigate the mean tardiness of PFSP is through the use of Hybrid Ant Lion Optimization (HALO). Even though ALO has been reported to be used to solve some scheduling problems, no studies have been found to have used HALO on PFSP to determine the mean tardiness performance. Therefore, the objective of this research was to develop an effective Hybrid Ant Lion Optimization (HALO) to solve the mean tardiness in PFSP. This involved the provision of HALO with an improved NEH-EDD algorithm developed by Kim [8] as a search agent and the use of computational experiments for analysis. These were conducted to establish a comparison with several other algorithms in previous literature.

This study contributes to the body of knowledge by 1) developing a new HALO algorithm to solve PFSP using mean tardiness performance, 2) determining the best parameters of the HALO algorithm, 3) discovering the computational time and efficiency of the HALO algorithm compared to previous ones. Therefore, the Part 2 of this paper explains the assumption and description of problems, formulation of problems, proposed algorithm, data generated, and experimental procedure, Part 3 presents experimental parameters and benchmarking algorithms while the inference is presented in Part 4.

2. Method

2.1. Problem assumption and description

The assumptions made in the problem are (1) sequencing involves the collection of n tasks which are ready at time 0, (2) each job has processing times for each process, (3) the time available for resources or machines is 0, (4) the set-up time is zero and covered by the processing time, (5) jobs are independent of each other, (6) interrupt is not allowed on every job, (7) each job has a fixed due date, (8) the machine only processes one job at a time, and (9) the processing time for every job on each machine is deterministic.

Table 1 provides an example of the problem for three machines and three jobs while Fig. 1 shows the completion time of each job if the sequences J1, J3, and J2 are 6, 9, and 11 in (a) and if J2, J3, and J1 are 6, 9, and 11 respectively in (b). Even though the two schedules have the same time of completion, sequence (b) had the best result for the mean tardiness with a value of 0 while (a) had 1.65 hours with a due date of 11,9 and 6. This means (a) has 5-hour tardiness while (b) has none.

Table 1. Time of process, dj at every machine (hour)

jobs	Machines			dj
	M1	*M2*	*M3*	
J1	3	2	1	11
J2	3	1	2	6
J3	2	1	3	9

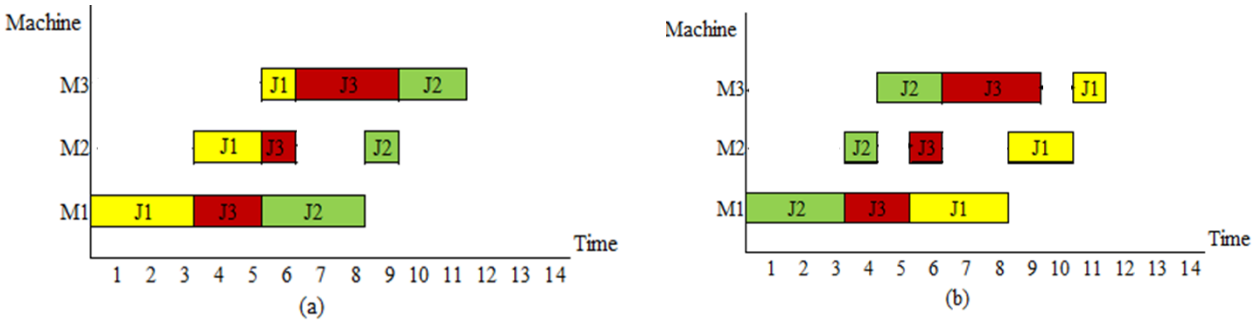


Fig. 1. The difference of 2 sequences

The aim was to minimize mean tardiness, and the notation used throughout this paper is shown in Table 2.

Table 2. The notation used throughout this paper

Variable	Definition
i	Jobs Index i $= 1,2 \ldots, n$
j	Machine index, j $= 1,2 \ldots, m$
n	Total number of jobs
m	Total number of machines
Pi,j	Processing time of job sequence i on machines j
Ci,j	Completion time of job sequence i on machines j
Ci	Completion time job i
Ti	Tardiness for a job i
$\bar{T}$	Mean tardiness
L	Total lateness.

Therefore, using these notations, the following formulas were developed from the PFSP problem:

$$C_{1,1} = P_{1,1} \quad (1)$$

$$C_{1,j} = C_{1,j-1} + P_{1,j}\ ,\ j = 2..m \quad (2)$$

$$C_{i,1} = C_{i-1,1} + P_{i-1,1}\ ,\ i = 2..n \quad (3)$$

$$C_{i,j} = max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j}\ , i = 2..n, j = 2..m \quad (4)$$

$$Ci = max(C_{i,j})\ ,\ \forall\, i = 1..n,\ j = 1..m \quad (5)$$

$$Ti = max(Ci - di, 0), \forall\, j, 1 \leq j \leq n \quad (6)$$

$$\bar{T} = \left(\sum_{i=1}^{n} T_i\right)/n \quad (7)$$

$$Li = Ci - di \quad \forall\, i, 1 \leq i \leq n \quad (8)$$

$$L = \sum_{i=1}^{n} L_i \quad (9)$$

The best permutations are defined as the ones with minimal mean tardiness. Therefore, the PFSP model to minimize tardiness is as follows:

$$\text{Objective function Z} = \min \sum_{i=1}^{n} T_i)/n \quad (10)$$

Subject to:

$$\left.\begin{aligned} & C_{1,1} = P_{1,1} \\ & C_{1,j} = C_{1,j-1} + P_{1,j}\,, j = 2..m \\ & C_{i,1} = C_{i-1,1} + P_{i-1,1}\,, i = 2..n \\ & C_{i,j} = max\left(C_{i-1,j}, C_{i,j-1}\right) + P_{i,j}\,, i = 2..n, j = 2..m \\ & Ci = max(C_{i,j})\ ,\ \forall\, i = 1..n,\ j = 1..m \\ & Ti = max(Ci - di, 0), \forall\, j, 1 \le j \le n \end{aligned}\right\} \quad (11)$$

Equation (1) describes the completion time of sequence 1 on machine 1, Equation (2) on the completion time of sequence 1 on machines 2 to m, and Equation (3) on the completion time of Sequence i on the machine 1. Moreover, Equation (4) shows the completion time of sequence i on machines j, Equation (5) presents the completion time job i, Equation (6) shows the tardiness of the job i, Equation (7) shows the mean tardiness of the permutation, and Equation (8) describes the lateness job i. Furthermore, Equation (9) describes the total lateness of the permutation, Equation (10) explains the objective function of the PFSP model to minimize mean tardiness, and Equation (11) defines the constraint of the mean tardiness of the model. However, the constraints of the model are equations (1) to (6).

2.2. The Proposed Hybrid Ant Lion Algorithm (HALO)

The Hybrid Ant Lion Algorithm (HALO), which mimics the survival behavior of Lion and Ant, was proposed. This was an advancement to the research conducted by Mirjalili [26], which proposed the Ant Lion Algorithm (ALO) to solve the problem of continuous optimization without the function to complete the PFSP. This method made use of local search strategies such as swaps, flips, and slides and with the insertion of the NEH algorithm (Algorithm 1) to replace a one search agent to solve the PFSP problem. HALO consisted of five steps which include initializing the search agent position, converting the position to Large Rank Value (LRV) job permutation, changing the position of one search agent using NEH, updating the position of the search agent, and performing a local swap search, flip, and slide search. Moreover, the HALO pseudo-code was shown in Algorithm 2. The five steps are, however, discussed in the following sections.

2.2.1. Initialization of the search agent positions

The initial location of the search agent was randomly generated and configured to avoid no loop numbers in similar agents [15], as shown in Fig. 2. The matrix shows the number of search agents in the row and the dimensions in the column based on the number of jobs. However, Fig. 2 shows the matrix with 2 rejected position has the same random number for search agent 1 with 0.81 and search agent 2 with 0.43. Therefore, it cannot be used for scheduling.

(1) Accepted position	(2) Rejected position
$\begin{bmatrix} 0.12 & 0.29 & 0.11 & 0.81 \\ 0.61 & 0.43 & 0.88 & 0.76 \\ 0.55 & 0.72 & 0.64 & 0.94 \end{bmatrix}$	$\begin{bmatrix} 0.12 & 0.33 & 0.81 & 0.81 \\ 0.61 & 0.82 & 0.43 & 0.43 \\ 0.55 & 0.76 & 0.72 & 0.94 \end{bmatrix}$

Fig. 2. Initialization of the positions of the search agent

Ant position is presented with a M_{Ant} matrix (Equation (12)).

$$M_{Ant} = \begin{bmatrix} A1.1 & A1.2 & \cdots & A1,d \\ A2.1 & A2.2 & \cdots & A2,d \\ \vdots & \vdots & \ddots & \vdots \\ An.1 & An.2 & \cdots & An,d \end{bmatrix} \quad (12)$$

The position Ant matrix was represented by M_{Ant}. For $A_{n,d}$, d indicates the n-th Ant vector while n is the number of Ants. Moreover, the fitness function of the Ant was used as the matrix M_{OA} based on (13).

$$M_{OA} = \begin{bmatrix} f([A1,1\,.A1,2\,,\ldots,A1,d]) \\ f([A2,1\,.A2,2\,,\ldots,A2,d]) \\ \vdots \\ f([An,1\,.An,2\,,\ldots,An,d]) \end{bmatrix} \quad (13)$$

$M_{\text{Ant Lion}}$ describes the position matrix for the Ant Lion according to (14) while the fitness function is represented by the matrix M_{OAL} based (15).

$$M_{\text{Ant Lion}} = \begin{bmatrix} AL1,1 & AL1,2 & \ldots & AL1,d \\ AL2,1 & AL2,2 & \ldots & AL2,d \\ \vdots & \vdots & \ddots & \vdots \\ ALn,1 & ALn,2 & \ldots & ALn,d \end{bmatrix} \quad (14)$$

$$M_{\text{OAL}} = \begin{bmatrix} f([AL1,1\,.AL1,2\,,\ldots,AL1,d]) \\ f([AL2,1\,.AL2,2\,,\ldots,AL2,d]) \\ \vdots \\ \vdots \\ f([ALn,1\,.ALn,2\,,\ldots,ALn,d]) \end{bmatrix} \quad (15)$$

For the $AL_{n,d}$, d indicates the n-th Ant Lions' vector, n is the number of Ant Lions while f is the objective function.

2.2.2. The NEH-EDD algorithm

This study made use of a new NEH-EDD algorithm as a search agent, and simple NEH usually involves setting the number of processing times for each job from the largest to the smallest [10]. However, in Kim [8] NEH-EDD algorithm, the work sequence was based on the smallest due date to the largest and was calculated to have the best mean tardiness. Therefore, a new NEH-EDD algorithm was proposed to minimize mean tardiness differently from the NEH. This involves sorting the jobs by the smallest due date followed by the evaluation of the subsequences by the mean tardiness and total lateness as shown in Algorithm 1 (Fig. 3).

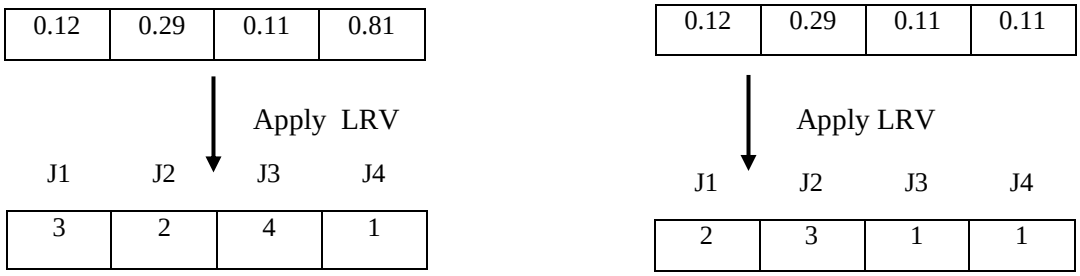
Algorithm 1. Proposed NEH-EDD Procedure

1. Input: S = tasks arranged in the increase order of dj, where dj = d1≤ d2≤...≤ dj
2. Select the partial sequence with the smallest mean tardiness using Equation (8) into the partial sequence of ties between σ = ($S_{[1]}$, $S_{[2]}$) or σ = ($S_{[2]}$, $S_{[1]}$), and if the mean tardiness is the same, the partial sequence with the minimum total lateness using (9) should be selected.
3. **for** k = 3 **to** n **do**
4. Investigate the insertion of $S_{[k]}$ in every feasible place in σ from 1 to k + 1 in the partial sequence σ.
5. Select the best insertion, for instance, the insertion with the smallest mean tardiness using (7) and if the mean tardiness is the same, select the partial sequence with the minimum total lateness calculated from (9).
6. **end for**
7. **return** (s)

Fig. 3. Proposed NEH-EDD Procedure algorithm.

2.2.3. Apply Large Rank Value (LRV) to convert search agents to job permutations

The use of Large Rank Value (LRV) was proposed for the conversion of search agents to job permutations. LRV is an effective method to map continuous values into job permutations by ranking the values from the largest to the smallest [31]. Fig. 4(a) shows the incorrect job permutations generated due to the PFSP condition, and Fig. 4 (b) is also not correct because it has the same position as the value. Therefore, the first job is in two different types.



(a) Correct Job Permutation (b) Incorrect Job Permutation

Fig. 4. Large Rank Value Representation

2.2.4. Update Position of Ant Lion Algorithm (ALO)

Ant's Lion was allowed to hunt other Ants down when moving around looking for food. This concept is, therefore, modeled in (16).

$$X(t) = [0, cumsum\,(2r(t_1) - 1), cumsum\,(2r(t_2) - 1), \dots, cumsum\,(2r(t_N) - 1] \quad (16)$$

Cumsum is a calculation of sum, N is the maximum number of iterations, and t indicates the t-iteration. Moreover, $r(t)$ was formulated as a stochastic function in (17).

$$r(t) = \begin{cases} 1, & if\ rand > 0{,}5 \\ 0, & if\ rand \leq 0{,}5 \end{cases} \quad (17)$$

The Ant Lion moves in a random position in each step, and Equation (18) was used to ensure the Ant Lion's journey remains within the search range [21].

$$X_i^t = \frac{(X_i^t - a_i)\ X\ (d_i - c_i^t)}{(d_i^t - a_i)} + c_i\ , \quad (18)$$

Where a_i is the minimum random walk of a $i - th$ variable, c_i is the maximum random walk in the i variable, d_i^t shows the maximum i variable in the iteration $t - th$, and c_i^t is the minimum i variable in the $t - th$ iteration. Moreover, the traps on the Ant Lion were modeled, as shown in the (19).

$$c_i^t = Antlion_j^t + c^t,\ d_i^t = Antlion_j^t + d^t \quad (19)$$

Where c^t is the minimum of all variables in the $t - th$ iteration, d^t indicates the vector of the maximum variables in the $t - th$ iteration, C_j^t is the minimum of all variables for the $i - th$ Ant, d_j^t is the maximum of all variables for Ants $i - th$, and $Antlion_j^t$ shows the position of the $j - th$ Ant Lion selected at the $t - th$ iteration. Furthermore, the shift of the Ants to the directions of the Ant Lion was modeled on (20) to produce

$$c^t = \frac{c^t}{I},\ d^t = \frac{d^t}{I},\ I = 10^w \cdot \frac{t}{T} \quad (20)$$

Each Ant runs randomly around the Ant Lion chosen by the roulette wheel, and the position of the Ant Lion was selected in the Iteration using (21).

$$Ant_m^{It} = \frac{R_l^{It} + R_e^{It}}{2} \quad (21)$$

Where Ant_m^{It} describes the position of the Ant Lion selected in It iteration, R_l^{It} shows a random walk around the Ant Lion selected from the roulette wheel in the $It-th$ iteration, and R_e^{It} describes a random walk around the elite in $It-th$ iteration. In the ALO algorithm, the Ant Lion changes its position to the hunted position if Ant has better fitness, and this is represented in (22).

$$Antlion_l^{It} = Ant_m^{It} \text{ if } f(Ant_m^{It}) > f(Antlion_l^{It}) \quad (22)$$

2.2.5. The local search

The local search is a combinatorial method applied to optimize the initial sequence up to when the optimum objective function is obtained [32], and the best solution is produced. Effective metaheuristic algorithms generally apply to local search [33], and the proposed steps include swap, flip, and slide. Do swap was conducted by randomly exchanging two job sequences, and the transactions in the iterations to t were repeated as $n(n-1)/2$ as illustrated in Fig. 5. The Do Flip rule was to reverse the order of the selected jobs, and the operation in iterations to t was repeated as $n(n-1)/2$ as illustrated in Fig. 6. Moreover, Do Slides were conducted by shifting the job sequence and the operation in the iteration t was also repeated as much as n^2 as shown in Fig. 7. However, they are not allowed to be repeated in one iteration in the proposed local search.

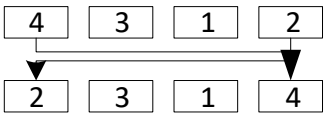


Fig. 5. Do Swap Illustration

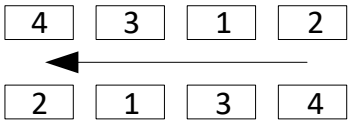


Fig. 6. Do Flip Illustration

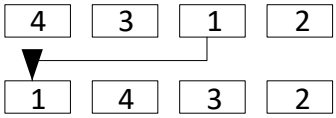


Fig. 7. Do Slide Illustration

2.3. The data generation and experimental procedure

Several experiments were randomly conducted to evaluate the proposed HALO algorithm (Fig. 8) through the use of data from the number of jobs, the number of phases (machines), time of operation, and the due date. The experiments were conducted in three-set groups include small (5, 12 and 19), moderate (26, 30 and 50), and large (75, 100, and 200), and this led to the difference of 9 in the overall number of jobs. For the purpose of this study, four different jobs, comprising of 5, 7, 10, and 20, were used with three variations of machines in each group. Moreover, processing time was generated by a uniform distribution of 12, 497 in minutes units while the due date for each job was randomly generated with a uniform distribution of 2100, 5040 in minutes. Experiments were also conducted with a combination of different parameters in the HALO algorithm through the use of variables of population and iteration. However, the population analysis made use of 2 levels, including 5 and 20, while the iterative experimental parameters used 9 levels, including 10, 15, 20, 25, 30, 35, 40, 45, and 50. It is also important to state that 162 experiments were conducted covering each group to ensure the best parameters of the HALO algorithm.

Algorithm 2. Proposed Hybrid Ant Lion Algorithm

Input: a set Job, the due date, iteration maximum, population.
Randomly initialize the first population of Ants and Ant Lions
Apply LRV on each search agent to be mapped into job permutation
Solve the PFSSP using NEH
Choose one search agents from the population and replace it with NEH
Calculate the fitness of Ants and Ant Lions using Equation (8)
Find the best Ant Lions and assume it as the elite or determined optimum
W* = The best search agent
t=0
while (t < iteration)
    for every Ant (population)
        Select an Ant Lion using a Roulette wheel
        Update c and d using (20)
        Create a random walk and normalize it using (16) and (18)
        Update the position of Ant using (21)
    end for
  Apply LRV on each search agent to be mapped into job permutation
    Calculate the fitness of all Ants using (8)
    Replace an Ant Lion with its corresponding Ant if it becomes fitter using (22)
    W^t= best fitness in iteration t
 Update the elite if an Ant Lion becomes fitter than it.
  If $W^t < W^*$
    $W^* = W^t$
  End if
  Apply Local search
  For i = 1: $n(n-1)/2$
    Perform swap on the W^*. Ensure no repeated swap in the W^*
      If $W^i < W^*$
      $W^* = W^i$
      End if
   End for
  For j = 1: $n(n-1)/2$
    Perform flip on the W^*. Ensure no repeated swap in the W^*
      If $W^j < W^*$
      $W^* = W^j$
      End if
   End for
  For k = 1: n^2
    Perform slide on the W^*. Ensure no repeated swap in the W^*
      If $W^k < W^*$
      $W^* = W^k$
      End if
   End for
end while
Return elite

Fig. 8. Proposed Hybrid Ant Lion Algorithm

Furthermore, to investigate the effectiveness of the HALO, its best algorithm parameters were compared with several previous algorithms such as Kim [8], Vallada and Ruiz [16], Parthasarathy and Rajendran [34], Nagano *et al.* [11], and ALO [30]. Moreover, the computational experiments were written in MATLAB R16 software for Windows 8.1 AMD x86-64 RAM 4 GB. The performance was measured using the Efficiency Index Percentage (EIP) and Relative Error Percentage (REP). The EIP was identified as the mean tardiness ratio between the HALO and another percentage algorithm in (21),

and HALO was considered more powerful If EIP < 100 percent, the same if it is equal to 100%, and less powerful if EIP > 100 percent. The REP was also tested with another algorithm using (22).

$$\text{EIP} = \frac{\overline{T}_{\text{proposed algorithm}}}{\overline{T}_{\text{another algorithm}}} \text{x} 100\% \quad (21)$$

$$\text{REP} = \frac{\overline{T}_{\text{another algorithm}} - \overline{T}_{\text{proposed algorithm}}}{\overline{T}_{\text{proposed algorithm}}} \text{x} 100\% \quad (22)$$

3. Results and Discussion

3.1 Experimental parameters of the Hybrid Ant Lion Optimization

The result of the simulated parameters of HALO is shown in Table 3, and it was discovered that a higher number of iterations and population led to lower mean tardiness. This is in accordance with the claims of Haddock and Mittenthal [24]. Furthermore, the computational time simulation parameters were shown in Table 4, and the results showed a higher number of iterations and populations produced significant computational time, which further led to a better selection of a solution. Moreover, a higher number of jobs was found to require more time. Therefore, the PFSP was included in the NP-Hard problems in line with the result of the research conducted by Utama *et al.* [7] and Garey *et al.* [35].

Table 3. Comparison of each parameter towards efficient mean tardiness

Iteration	Job Family	n x m	Population								
			10	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
5	*Small*	5 x 5	0	0	0	0	0	0	0	0	0
		12 x 7	798	757	704	667	663	661	650	638	632
		19 x 10	1801	1584	1544	1442	1431	1393	1369	1340	1329
	Medium	26 x 20	2362	2251	2239	2204	2162	2156	2113	2098	2097
		30 x 5	2669	2579	2533	2436	2423	2389	2389	2372	2336
		50 x 10	5111	5099	5090	5080	5030	5004	4969	4913	4869
	High	75 x 20	8417	8349	8293	8281	8225	8215	8175	8166	8053
		100 x 5	11546	11499	11483	11473	11469	11416	11317	11294	11234
		200 x 10	23848	23683	23559	23445	23265	23262	23206	23171	23134
20	*Small*	5 x 5	0	0	0	0	0	0	0	0	0
		12 x 7	716	648	641	639	629	628	617	610	610
		19 x 10	1499	1429	1417	1379	1371	1362	1336	1305	1289
	Medium	26 x 20	1381	1378	1372	1294	1270	1262	1240	1206	1162
		30 x 5	2331	2325	2169	2156	2145	2107	2007	1999	1950
		50 x 10	5115	4918	4862	4822	4743	4736	4679	4673	4610
	High	75 x 20	7950	7817	7806	7615	7424	7376	7375	7365	7313
		100 x 5	10922	10898	10883	10785	10773	10769	10682	10609	10549
		200 x 10	23904	23667	23453	23374	23257	22982	22945	22899	22690

Table 4. Comparison of each parameter towards computational time

Iteration	Job Family	n x m	Population								
			10	*15*	*20*	*25*	*30*	*35*	*40*	*45*	*50*
5	*Small*	5 x 5	0.000	0.016	0.016	0.031	0.031	0.031	0.031	0.031	0.047
		12 x 7	0.000	0.016	0.027	0.031	0.047	0.047	0.063	0.063	0.094
		19 x 10	0.047	0.063	0.063	0.063	0.078	0.094	0.094	0.109	0.125
	Medium	26 x 20	0.094	0.109	0.109	0.141	0.172	0.203	0.203	0.234	0.266
		30 x 5	0.218	0.218	0.219	0.234	0.250	0.266	0.266	0.281	0.313
		50 x 10	0.594	0.672	0.672	0.672	0.703	0.719	0.750	0.750	1.156
	High	75 x 20	1.875	1.922	1.969	1.984	2.016	2.109	2.125	2.188	2.234
		100 x 5	4.547	4.703	4.734	4.766	4.781	4.828	4.984	5.031	5.250
		200 x10	43.54	43.547	43.750	43.813	43.82	43.98	44.094	44.17	44.34
20	*Small*	5 x 5	0.000	0.047	0.034	0.085	0.085	0.066	0.088	0.069	0.103
		12 x 7	0.000	0.047	0.065	0.081	0.127	0.094	0.175	0.156	0.272
		19 x 10	0.113	0.169	0.125	0.175	0.187	0.253	0.253	0.252	0.325
	Medium	26 x 20	0.281	0.306	0.328	0.380	0.378	0.528	0.609	0.609	0.770
		30 x 5	0.502	0.546	0.635	0.656	0.675	0.691	0.531	0.563	0.844
		50 x 10	1.188	1.344	1.949	1.478	1.687	1.438	1.650	1.875	3.006
	High	75 x 20	3.750	5.766	4.725	4.167	4.636	4.852	6.163	5.031	4.916
		100 x 5	12.27	10.347	10.416	12.391	11.47	11.10	12.46	11.57	12.600
		200 x10	87.23	121.93	131.25	131.43	92.03	92.36	110.23	88.34	124.16

3.2 Results Comparison Algorithms

As previously stated, the HALO was compared with several algorithms, and it was discovered that while the number of cases was increasing, a better result of mean tardiness was recorded as shown in Table 5. It also showed HALO needed a large computational time for each case study to produce better solutions.

Table 5. The comparison of mean tardiness and computational time (in second)

Job Family	n x m	HALO		NEH-EDD [8]		Nagano *et al.* [11]		GA [16]		SA [34]		ALO [30]	
		Mean Tardiness	*Time*	*Mean Tardiness*	*Time*	*Mean Tardiness*	*Time*	*Mean Tardiness*	*Time*	*Mean Tardiness*	*Time*	*Mean Tardiness*	*Time*
Small	5 x 5	0	0.103	0	0.000	0	0.000	0	0.104	0	0.047	0	0.000
	12 x 7	609	0.272	726	0.012	639	0.083	673	0.154	649	0.094	700	0.083
	19 x 10	1289	0.325	1568	0.125	1405	0.119	1386	0.113	1382	0.125	1477	0.119
Medium	26 x 20	1162	0.770	2252	0.294	1295	0.288	1301	0.281	2113	0.266	1777	0.288
	30 x 5	1950	0.844	2579	0.318	2117	0.410	2171	0.502	2399	0.313	2375	0.410
	50 x 10	4610	3.006	5131	0.594	4777	0.891	4758	1.188	4933	1.156	4945	0.891
High	75 x 20	7313	4.916	8309	1.875	7415	2.813	7468	3.750	8206	2.234	7889	2.813
	100 x 5	10549	12.600	11525	4.547	10812	8.412	10795	12.277	11306	5.250	11160	8.412
	200 x 10	22690	124.163	23581	43.542	23005	65.388	23275	87.234	23208	44.344	23428	65.388

Table 6 shows the comparison between the Relative Error Percentage (REP) and Efficiency Index Percentage (EIP) of the HALO and several algorithms. Job family including small, medium, and large sizes was used for evaluation. The result showed the REP of HALO was stronger than other algorithms, such as NEH-EDD [8] with 23%, Nagano *et al.* [11] with 5%, Genetic Algorithm [16] 6%, Simulated Annealing 16%, and ALO 10%. It was also discovered that as the number of cases increased, the HALO provided better performance over the other four algorithms. Moreover, the Efficiency Index Percentage (EIP) revealed the increment in the number of cases led to the provision of more significant performance for the HALO algorithm. The values of EIP of mean tardiness for NEH-EDD [8], Nagano *et al.* [11],

Genetic [16], Simulated Annealing, and ALO were found to be 23%, 84%, 96%, 95%, and 89% respectively. Therefore, all the numerical experiments conducted showed HALO algorithm performs better.

Table 6. Comparison of Relative Error Percentage (REP) and Efficiency Index Percentage (EIP)

Job Family	n x m	NEH-EDD [8]		Nagano *et al.* [11]		GA [16]		SA [34]		ALO [30]	
		EIP	*REP*	*EIP*	*REP*	*EIP*	*REP*	*EIP*	*REP*	*EIP*	*REP*
Small	5 x 5	100%	0%	100%	0%	100%	0%	100%	0%	100%	0%
	12 x 7	84%	19%	95%	5%	90%	11%	87%	7%	87%	12%
	19 x 10	82%	22%	92%	9%	93%	8%	87%	7%	87%	12%
Medium	26 x 20	52%	94%	90%	11%	89%	12%	65%	82%	65%	27%
	30 x 5	76%	32%	92%	9%	90%	11%	82%	23%	82%	16%
	50 x 10	90%	11%	97%	4%	97%	3%	93%	7%	93%	7%
High	75 x 20	88%	14%	99%	1%	98%	2%	93%	12%	93%	7%
	100 x 5	92%	9%	98%	2%	98%	2%	95%	7%	95%	5%
	200 x 10	96%	4%	99%	1%	97%	3%	97%	2%	97%	3%

4. Conclusion

The problem of permutation flow shop scheduling was studied to minimize mean tardiness through the use of Hybrid Anti-Lion Optimization (HALO) algorithms, and the performance was compared with several other algorithms. The result of the computational experiments conducted showed the Hybrid Ant-Lion optimization algorithm achieved optimum mean tardiness. It is recommended that this approach is used as an initial solution for different metaheuristic algorithms and also in reducing the mean tardiness in more complicated permutation flow shop scheduling problems.

Acknowledgment

The authors appreciate the Directorate of Research at the University of Muhammadiyah Malang for the support provided for the research as well as the Department of Industrial Engineering Optimization Laboratory for the use of their facilities.

References

[1] D. M. Utama, T. Baroto, D. Maharani, F. R. Jannah, and R. A. Octaria, "Algoritma Ant-Lion optimizer untuk meminimasi emisi karbon pada penjadwalan flow shop dependent sequence set-up," *2019,* vol. 9, pp. 69-78, 2019-06-28 2019, doi: 10.24960/jli.v9i1.4775.69-78.

[2] M. S. Nagano, R. Ruiz, and L. A. N. Lorena, "A Constructive Genetic Algorithm for permutation flowshop scheduling," *Computers & Industrial Engineering,* vol. 55, pp. 195-207, 2008, doi: 10.1016/j.cie.2007.11.018.

[3] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flow shop scheduling under make to stock production system," *Computers & Industrial Engineering,* vol. 90, pp. 12-24, 2015, doi: 10.1016/j.cie.2015.08.006.

[4] Q. C. Ta, J.-C. Billaut, and J.-L. Bouquard, "Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem," *Journal of Intelligent Manufacturing,* vol. 29, pp. 617-628, 2018, doi: 10.1007/s10845-015-1046-4.

[5] D. M. Utama, L. R. Ardiansyah, and A. K. Garside, "Penjadwalan Flow Shop untuk Meminimasi Total Tardiness Menggunakan Algoritma Cross Entropy–Algoritma Genetika," *Jurnal Optimasi Sistem Industri,* vol. 18, pp. 133-141, 2019, doi: 10.25077/josi.v18.n2.p133-141.2019.

[6] K. Karabulut, "A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops," *Computers & Industrial Engineering,* vol. 98, pp. 300-307, 2016, doi: 10.1016/j.cie.2016.06.012.

[7] D. M. Utama, D. S. Widodo, W. Wicaksono, and L. R. Ardiansyah, "A New Hybrid Metaheuristics Algorithm for Minimizing Energy Consumption in the Flow Shop Scheduling Problem," *International Journal of Technology,* vol. 10, pp. 320-331, 2019, doi: 10.14716/ijtech.v10i2.2194.

[8] Y.-D. Kim, "Heuristics for flowshop scheduling problems minimizing mean tardiness," *Journal of the Operational Research Society,* vol. 44, pp. 19-28, 1993, doi: 10.1057/jors.1993.3.

[9] V. Fernandez-Viagas and J. M. Framinan, "NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness," *Computers & Operations Research,* vol. 60, pp. 27-36, 2015, doi: 10.1016/j.cor.2015.02.002.

[10] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega,* vol. 11, pp. 91-95, 1983, doi: 10.1016/0305-0483(83)90088-9.

[11] M. S. Nagano, F. L. Rossi, and C. P. Tomazella, "A new efficient heuristic method for minimizing the total tardiness in a no-idle permutation flow shop," *Production Engineering,* vol. 11, pp. 523-529, 2017, doi: 10.1007/s11740-017-0747-2.

[12] V. Fernandez-Viagas and J. M. Framinan, "Efficient non-population-based algorithms for the permutation flowshop scheduling problem with makespan minimisation subject to a maximum tardiness," *Computers & Operations Research,* vol. 64, pp. 86-96, 2015, doi: 10.1016/j.cor.2015.05.006.

[13] R. M'Hallah, "An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop," *International Journal of Production Research,* vol. 52, pp. 3802-3819, 2014/07/03 2014, doi: 10.1080/00207543.2014.899719.

[14] J.-M. Kim, Y.-D. Zhou, and D.-H. Lee, "Priority scheduling to minimize the total tardiness for remanufacturing systems with flow-shop-type reprocessing lines," *The International Journal of Advanced Manufacturing Technology,* vol. 91, pp. 3697-3708, 2017, doi: 10.1007/s00170-017-0057-z.

[15] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem," *Future Generation Computer Systems,* vol. 85, pp. 129-145, 2018, doi: 10.1016/j.future.2018.03.020.

[16] E. Vallada and R. Ruiz, "Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem," *Omega,* vol. 38, pp. 57-67, 2010, doi: 10.1016/j.omega.2009.04.002.

[17] V. c. A. Armentano and D. P. Ronconi, "Tabu search for total tardiness minimization in flowshop scheduling problems," *Computers & Operations Research,* vol. 26, pp. 219-235, 1999, doi: 10.1016/S0305-0548(98)00060-4.

[18] H. Mokhtari and A. Noroozi, "An efficient chaotic based PSO for earliness/tardiness optimization in a batch processing flow shop scheduling problem," *Journal of Intelligent Manufacturing,* vol. 29, pp. 1063-1081, 2018, doi: 10.1007/s10845-015-1158-x.

[19] S. Hasija and C. Rajendran, "Scheduling in flowshops to minimize total tardiness of jobs," *International Journal of Production Research,* vol. 42, pp. 2289-2301, 2004, doi: 10.1080/00207540310001657595.

[20] W. Shao, D. Pi, and Z. Shao, "A hybrid discrete teaching-learning based meta-heuristic for solving no-idle flow shop scheduling problem with total tardiness criterion," *Computers & Operations Research,* vol. 94, pp. 89-105, 2018, doi: 10.1016/j.cor.2018.02.003.

[21] W. Liao and Y. Fu, "A New Robust Scheduling Model for Permutation Flow Shop Problem," *Recent Advances in Intelligent Manufacturing,* pp. 308-317, 2018, doi: 10.1007/978-981-13-2396-6_29.

[22] R. Nasution, A. K. Garside, and D. M. Utama, "Penjadwalan Job Shop Dengan Pendekatan Algoritma Artificial Immune System," *2017,* vol. 18, p. 14, 2017, doi: 10.22219/JTIUMM.Vol18.No1.29-42.

[23] J.-n. Shen, L. Wang, and S.-y. Wang, "A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion," *Knowledge-Based Systems,* vol. 74, pp. 167-175, 2015, doi: 10.1016/j.knosys.2014.11.016.

[24] J. Haddock and J. Mittenthal, "Simulation Optimization Using Simulated Annealing," *Computers & Industrial Engineering,* vol. 22, pp. 387-395, 1992, doi: 10.1016/0360-8352(92)90014-B.

[25] D. M. Utama, "An Effective Hybrid Sine Cosine Algorithm to Minimize Carbon Emission on Flow-shop Scheduling Sequence Dependent Setup," *2019,* vol. 20, pp. 62-72, 2019, doi: 10.22219/JTIUMM.Vol20.No1.62-72.

[26] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software,* vol. 83, pp. 80-98, 2015, doi: 10.1016/j.advengsoft.2015.01.010.

[27] N. Chopra and S. Mehta, "Multi-objective optimum generation scheduling using Ant Lion Optimization," in *2015 Annual IEEE India Conference (INDICON)*, 2015, pp. 1-6, doi: 10.1109/INDICON.2015.7443839.

[28] H. M. Dubey, M. Pandit, and B. K. Panigrahi, "Hydro-thermal-wind scheduling employing novel Ant Lion optimization technique with composite ranking index," *Renewable Energy,* vol. 99, pp. 18-34, 2016, doi: 10.1016/j.renene.2016.06.039.

[29] E. Umamaheswari, S. Ganesan, M. Abirami, and S. Subramanian, "Deterministic reliability model based preventive generator maintenance scheduling using Ant Lion Optimizer," in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2016, pp. 1-8, doi: 10.1109/ICCPCT.2016.7530272.

[30] M. Petrović, J. Petronijević, M. Mitić, N. Vuković, Z. Miljković, and B. Babić, "The Ant Lion optimization algorithm for integrated process planning and scheduling," in *Applied Mechanics and Materials*, 2016, pp. 187-192, doi: 10.4028/www.scientific.net/AMM.834.187.

[31] X. Li and M. Yin, "An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure," *Advances in Engineering Software,* vol. 55, pp. 10-31, 2013, doi: 10.1016/j.advengsoft.2012.09.003.

[32] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research,* vol. 130, pp. 449-467, 2001, doi: 10.1016/S0377-2217(00)00100-4.

[33] G. Laporte, M. Gendreau, J. Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research,* vol. 7, pp. 285-300, 2000, doi: 10.1111/j.1475-3995.2000.tb00200.x.

[34] S. Parthasarathy and C. Rajendran, "A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs-a case study," *Production Planning & Control,* vol. 8, pp. 475-483, 1997, doi: 10.1080/095372897235055.

[35] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of operations research,* vol. 1, pp. 117-129, 1976, doi: 10.1287/moor.1.2.117.