Jurnal Teknik Industri
Vol. 22, No. 2, August 2021, pp. 224-235

The Moth-Flame Optimization Algorithm for Flow Shop Scheduling Problem with Travel Time

Ikhlasul Amallynda *, Bhisma Hutama
Department of Industrial Engineering, Universitas Muhammadiyah Malang, Indonesia
* Correspondence Author: ikhlasulamallynda@umm.ac.id

ARTICLE INFO

Article history
Received June 3, 2021
Revised August 13, 2021
Accepted August 27, 2021
Available Online August 31, 2021

Keyword
Makespan
Transportation Time
Flow Shop
Moth Flame Optimization Algorithm

ABSTRACT

This article examined the flow shop scheduling problem by considering the travel time between machines. The objective function of this problem was to provide a makespan. The Moth Flame Optimization (MFO) algorithm was proposed to solve the flow shop problem. The MFO experiment was carried out with a combination of iteration parameters and the population of the MFO algorithm to solve the flow shop scheduling problem. The computational results showed that MFO could produce a better solution than the actual scheduling method. Furthermore, the MFO Proposal Algorithm was able to reduce the makespan by up to 3%.

1. Introduction

Scheduling is allocating machines to process a set of jobs within a certain period [1, 2]. Job is a series of activities that require allocating resources over a certain period to be completed [3-6]. Some of the scheduling functions are to minimize processing time, subscription lead time, and inventory levels. In addition, the purpose of scheduling is to increase utilities such as machine facilities, labor, and equipment [7]. Flow shop production scheduling is the most popular investigated scheduling problem [8]. One of the objective functions of flow shop scheduling is to minimize makespan. This objective function is the time required to complete all jobs on a series of machines [9]. The minimum *makespan* value is needed so that the completion time is faster [10]. The company realizes the importance of turnaround time in increasing customer loyalty. Generally, these companies use the First Come First Serve (FCFS) rule because this rule is very easy [11]. Scheduling is prepared by considering the various constraints that exist in the company. Proper scheduling has a good impact, such as minimizing makespan [12, 13].

There have been many studies on the implementation of metaheuristic algorithms on scheduling problems to minimize makespan. Some of these algorithms are Cross Entropy-Genetic Algorithm [14], Particle Swarm Optimization [10], Simulated Annealing Algorithm [15, 16]. Other algorithms such as Ant Colony Optimization [17], Tabu Search [18], GRASP with fixed threads [19], Differential Evolution Algorithm [20], Artificial Immune System Algorithm [21]. Iterated Greedy Algorithm & Social Cognitive Algorithm [22], and Moth Flame Optimization [23] are also proposed to minimize the makespan. From the literature that discusses metaheuristic algorithms, most of the flow shop problems focused on minimizing makespan by considering processing time. Unfortunately, research on flow shop scheduling rarely considers travel time. In practice, travel time is often encountered because it requires transportation time between machines.

To the researchers' knowledge, we only found research by Hurink and Knust [24], which discussed the problem of flow shop scheduling with transportation time. Research on flow shop scheduling by considering travel time was scarce. This study proposed a modified Moth Flame Optimization (MFO) algorithm to solve the scheduling problem by considering travel time. Based on our literature study, there has been no research on implementing the MFO algorithm for flow shop scheduling by considering travel time. This algorithm was first proposed by Mirjalili [25] and has been applied in various engineering fields such as feature selection [26] and flow shop scheduling with the sequence-dependent setup [23]. This algorithm mimics the behavior of moths at night. The Moth flies at a fixed angle with the moon pointing [25]. This study aimed to solve the flow shop scheduling problem by considering travel time with the MFO algorithm. Travel time was based on vehicles transporting materials back and forth between machines [27, 28]. This model imitated the problem of pickup and delivery [29]. Transfer time is always associated with each delivery [30]. This research was a case study in a company in Indonesia. The results of this study are expected to have a significant impact on the company in minimizing the makespan.

2. Methods

2.1 Problem Assumption and Definition

This study introduced assumptions such as (1) All machines are ready at t=0; (2) setup and deletion time including processing time; (3) travel time is separate from processing time; (4) there is no preemption in this problem; (5) each machine stops when its last operation is completed; (6) processing time and travel time have been determined; (7) m stage flow shop is considered; (8) m-1 transporter is engaged to transfer from machine j to j-1; (9) Makespan is based on the completion time on the machine.. To solve this problem, the notations used are as follows:

Parameters:

t_{ij}	: Process time of job $i \in N$ in the machine of j $\in M$
T_{ij}	: Process time of job $i \in N$ from j machine to j+1 machine
R_{ij}	: Return time of job $i \in N$ from j+1 machine to j machine

Decision Variables:

y_{ij}	: 1, if i job is processed in j machine; 0, otherwise
x_{ik}	: 1, if k job k is processed before i; 0, otherwise (i < k)
C_{ij}	: Completion time job i in j machine
TT_{ij}	: Completion time of j transporter to send i job from j to j+1 machines.
C_{max}	: Makespan

For this problem, the mixed-integer linear programming model is as follows:

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

$$\sum_{i \in L} t_{i1} y_{i1} \le C_{i1} \qquad \forall i \in N \tag{2}$$

$$(C_{ij-1} + T_{ij-1}) y_{ij} \le TT_{ij-1} \qquad \forall i \in N, \forall j \in M: j \ge 2 \tag{3}$$

$$\sum_{i \in L} (C_{ij-1} + T_{ij-1}) y_{ij} \le C_{ij} - (TT_{ij-1} + R_{ij-1}) \qquad \forall i \in N, \forall j \in M: j \ge 2 \tag{4}$$

$$\sum_{i \in L} (C_{ij-1} + T_{ij-1}) y_{ij} \le C_{ij} - (TT_{ij-1} + R_{ij-1}) + D x_{ik} \forall i, k \in N : k > i, \forall j \in M: j \ge 2 \tag{5}$$

$$C_{ij} - (TT_{ij-1} + R_{ij-1}) + Dx_{ik} \leq D - \sum_{i \in L} (C_{ij-1} + T_{ij-1}) y_{ij} \quad \forall i,k \in N : k > i, \forall j \in M : j \geq 2 \quad (6)$$
$$C_{im} \leq C_{max} \quad \forall i \in N \quad (7)$$
$$\sum_{i \in L} y_{ij} = 1; \quad \forall i \in N, \forall j \in M \quad (8)$$
$$y_{ij} = y_{i,j+1} \quad \forall i \in N, \forall j \in M : j < m \quad (9)$$

For this problem, the mixed-integer linear programming model is as follows: The objective function of Equation (1) is to minimize the completion time (*makespan*). Constraint (2) ensures that a job cannot be completed earlier than the processing time on the first machine. Constraint (3) assures that the i job cannot be transferred from j to $j + 1$ machines while still processing on machine j. Constraint (4) confirms that the job cannot be processed on the machine if it processed the previous job and has not been transferred to machine j. Job permutations are defined by constraints (5) and (6) to avoid overlapping the same machine. Makespan among all jobs on the last machine is calculated using Equation (7). Equations (8) and (9) ensure that each j machine processes i job only once.

2.2 Proposed Procedure with MFO

In this study, the researchers proposed a new modified MFO-based algorithm. In MFO, the two determining variables were the moth population and their position. Because moths can fly and change their vector, the moth position can be represented in the matrix presented in Equation (10).

$$M_t = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & \dots & A_{2,d} \\ A_{n,1} & A_{n,2} & \dots & \dots & A_{n,d} \end{bmatrix} \quad (10)$$

Where n was the number of moths and d was the dimension (number of available variables). Thus, for all moths, it could be assumed that there were rules for evaluating and sorting fitness presented in Equation (11).

$$\mathrm{OM} = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \quad (11)$$

The fitness value was an objective value for each Moth. Therefore, each Moth was calculated its fitness value. One of the most critical variables in the MFO algorithm was flame. Therefore, it was calculated by the Flames Matrix that was ordered as in Equation (12).

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \dots & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \dots & \dots & F_{2,d} \\ F_{n,1} & m_{n,2} & \dots & \dots & F_{n,d} \end{bmatrix} \quad (12)$$

It was also assumed to store fitness for flames by calculating the flame number as in equation (13). It should be noted that moths and flames were essential and were considered as different variables. The difference was how they updated each iteration. Moths indicated agents searching around the search space. Furthermore, the flames were showing the best position obtained by the Moth. In other words, flames could be said to be pins/flags that moths aimed for when they got the best position. It would be updated when it arrived in the next best position. With this, the previous best position would not be lost.

$$\mathrm{OF} = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{13}$$

After the initialization stage, the P function (moving the Moth in the search space) was executed iteratively until the T function (according to the termination criteria is met) has a value of True. The P function was the main function that drove moths in the search space. The position of each Moth was updated based on the flames using equation (14).

$$M_i = S(M_i, F_j) \tag{14}$$

Since the MFO was inspired by the movement of moths, each Moth's position concerning the flames could be updated by equation (15). Where D_i denoted the $i-th$ distance for the $j-th$ flames. The constant that determines the shape of the logarithmic spiral is denoted as b, and t is a random number.

$$S(M_i, F_j) = D_i \, . \, e^{bt} . \cos(2\pi t) + F_j \tag{15}$$

The order of fitness for each agent D could be calculated based on Equation (16). Where $M_i$denoted the $i-th$ flames, F_j indicated the $j-th$ flames, and D_i denoted the $i-th$ moth distance for the $j-th$ flames.

$$D_i = |F_j - M_i| \tag{16}$$

The next step was to update the flames. In the MFO algorithm, different moth positions in the search space reduced the exploitation of the solution. Therefore, the mathematical formula for reducing the number of flames during an iterative is presented in Equation (17), where I represents the current iteration. The maximum number of flames was denoted as N, while the total number of iterations was formulated as T.

$$Flame\ no = round(N - I * \frac{N-1}{T}) \tag{17}$$

After the best position was found, the last step was to determine the best solution from the optimal moth flame position. In this study, to convert the position of the vector moth flame to the work order, the Large Rank Value (LRV) principle was applied. This procedure was effective for changing continuous variables to discrete [31]. The proposed algorithm can be seen in Algorithm 1.

2.3 Data and Experiment

Data processing time, travel time, and return transporter are presented in Table 1. In total, 12 jobs were completed in this problem. This study implemented a population of Moth (N) and iterations of 50, 250, and 500, respectively. This population and iteration were combined to produce the best parameters for job completion. This study also compared the proposed algorithm with the algorithm offered by Hurink and Knust [24]. The experimental results recorded the computation time and machine completion (makespan), transporter completion (return), total engine idle, and total idle transportation. Sensitivity analysis was also presented by changing the T_{i1}, R_{i1}, T_{i2}, dan R_{i2} variables. The entire algorithm was coded in MATLAB R2020b and ran on a computer with an Intel Core i3-6006U CPU at 2.0 GHz, 8Gb RAM, and 1 TB HDD on a Windows 7 system.

Algorithm 1. Pseudocode Moth-Flame Optimization (MFO) Algorithm

```
for i = 1 : n
  for j = 1 : d
    M(i,j) = (ub(i) – lb(i)) * rand() + lb(i);
  end
end
Apply LRV
OM = FitnessFunction(M);
Update flame no using Eq. (17)
OM = FitnessFunction(M);
If iteration = 1
        F = sort(M);
        OF = sort(OM);
else
        F = sort($M_{t-1}$, $M_t$);
        OF = sort($M_{t-1}$, $M_t$);
end
for i = 1 : n
    for j = 1 : d
        Update r and t
        Caculate D using Eq.(16) with respect to the
     Corresponding moth
        Update M(i,j) using Eqs. (14) and (15) with respect to
     the corresponding Moth
    end
end
```

Table 1. Data processing time, travel time, and return transporter

Job i	t_{i1}	t_{i2}	t_{i3}	T_{i1}	R_{i1}	T_{i2}	R_{i2}
1	880	660	220	6	4	4	3
2	800	600	200	4	2	7	2
3	760	570	190	7	3	4	4
4	240	180	60	5	3	7	4
5	520	390	130	7	3	4	2
6	720	540	180	4	2	7	4
7	280	210	70	6	4	7	3
8	240	180	60	7	4	6	2
9	200	150	50	5	2	5	4
10	720	540	180	4	3	6	4
11	560	420	140	4	4	5	3
12	400	300	100	6	4	5	2

3. Results and Discussion

3.1 Optimization with MFO

Fig. 1 projects the number of recapitulations of *makespan* (job completion on the machine) using the MFO algorithm from each population and iteration. The results indicated that the overall population and iteration experiments had the same duration of 6530 minutes. Furthermore, it showed that 6530 minutes was the most optimal makespan in this problem. To solve the 12 job problems, this research required 50 iterations and a population of 50.

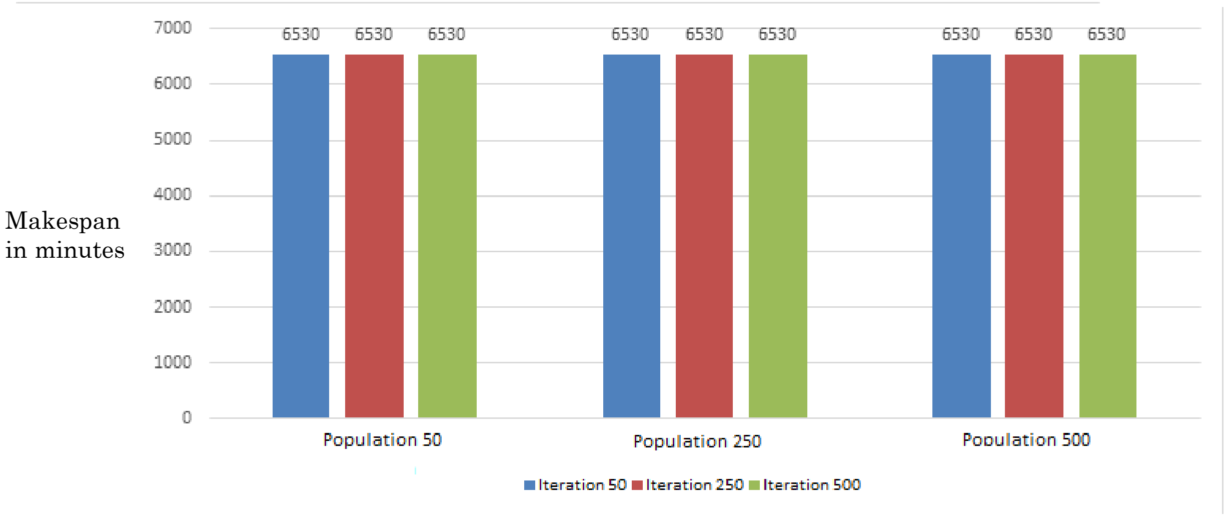

Fig. 1. MFO makespan recapitulation graph (in minutes)

The results of the MFO computation time to solve this problem are shown in Fig. 2. From the results in Fig. 2, it can be seen that the number of populations and iterations in problem-solving also affected the computation time. The larger the population and iterations, the greater the computational time generated, and vice versa. To solve the 12 job problems, the optimal computation time was 50 iterations and with 50 populations.

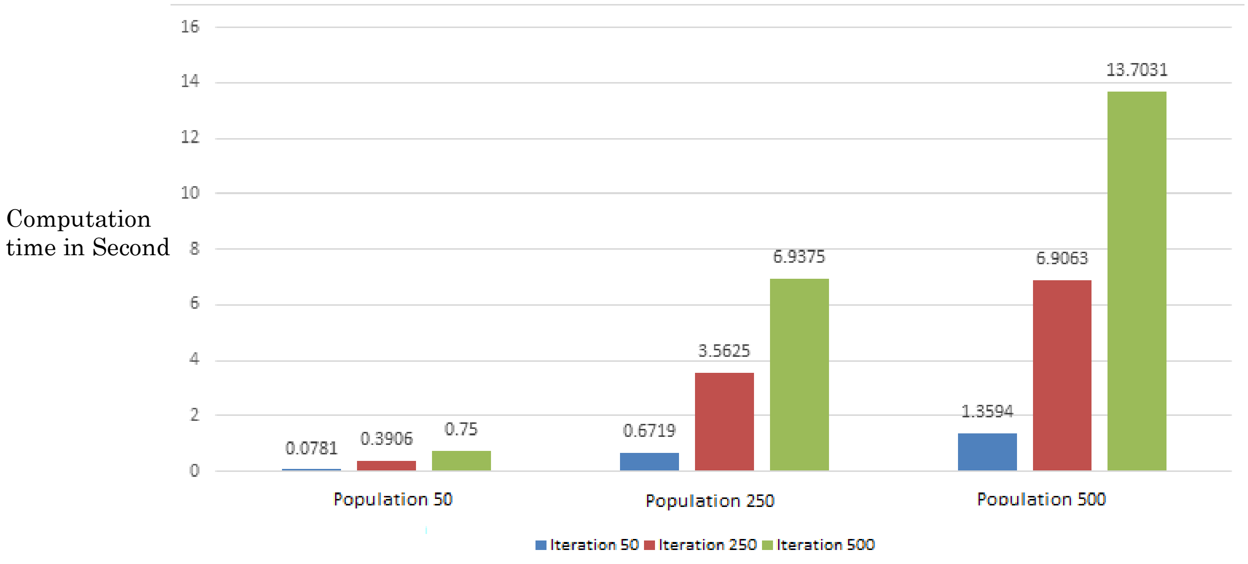

Fig. 2. MFO computing time (in Second)

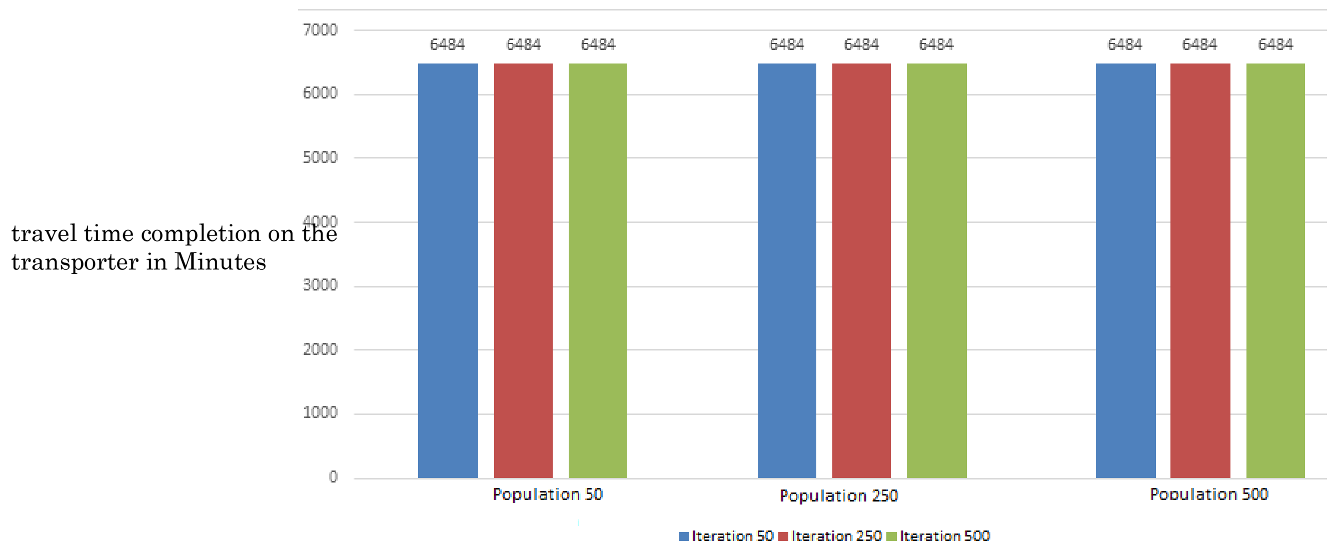

Fig. 3. Recapitulation of travel time completion on the transporter with MFO (minute)

The recapitulation of the completion of the travel time on the transporter with MFO in each experiment is shown in Fig. 3. The population and iteration had the same result, which was 6484 minutes. It was because of the population variation and iteration result solution optimal. Thus, variation in each population and iteration was not found. Based on MFO optimization, the optimal job order was 11-7-1-3-2-6-10-5-12-8-4-9.

3.2 Sensitivity Analysis

Sensitivity analysis is an analysis conducted to determine the effect of time changes on schedule. The experimental recapitulation on the change in T_{i1} is presented in Table 2. These results suggested that when T_{i1} was increased, the *makespan*, engine completion, completion transporter, Idle M1, Idle M2, Idle M3, Total idle M, Idle T2, Total Idle Transporter increased. Only idle transporter 1 (T1) and Total Idle T decreased. Conversely, when T_{i1} was decreased, then *makespan*, engine completion, transporter completion, Idle M1, Idle M2, Idle M3, Total idle M, Idle T2, Total Idle decreased. Only idle transporter 2 (T1) and Total Idle T increased.

Table 2. Recapitulation of experiments on changes in T_{i1}

T_{i1}	makespan	completion mesin	completion transporter	Idle M1	Idle M2	Idle M3	Total idle M	Idle T1	Idle T2	Total Idle T
5	6535	6535	6489	215	1795	4955	6965	6326	6385	12711
4	6534	6534	6488	214	1794	4954	6962	6337	6384	12721
3	6533	6533	6487	213	1793	4953	6959	6348	6383	12731
2	6532	6532	6486	212	1792	4952	6956	6359	6382	12741
1	6531	6531	6485	211	1791	4951	6953	6370	6381	12751
0	6530	6530	6484	210	1790	4950	6950	6381	6380	12761
-1	6529	6529	6483	209	1789	4949	6947	6392	6379	12771
-2	6528	6528	6482	208	1788	4948	6944	6403	6378	12781
-3	6527	6527	6481	207	1787	4947	6941	6414	6377	12791
-4	6526	6526	6480	206	1786	4946	6938	6425	6376	12801

A recapitulation of the experiment on changes in R_{i1} is presented in Table 3. These results showed that when R_{i1} was increased, the *makespan*, engine completion, completion transporter, Idle M1, Idle M2, Idle M3, Total idle M, Idle T2 were the same. Only idle transporter 1 (T1) and Total Idle T decreased. It was due to the very high processing time of the machine. Thus, the change in R_{i1} time did not significantly affect the completion.

Table 3. Recapitulation of experiments on changes in R_{i1}

R_{i1}	Makespan	Completion machine	Completion transporter	Idle M1	Idle M2	Idle M3	Total idle M	Idle T1	Idle T2	Total Idle T
3	6530	6530	6484	210	1790	4950	6950	6345	6380	12725
2	6530	6530	6484	210	1790	4950	6950	6357	6380	12737
1	6530	6530	6484	210	1790	4950	6950	6369	6380	12749
0	6530	6530	6484	210	1790	4950	6950	6381	6380	12761
-1	6530	6530	6484	210	1790	4950	6950	6393	6380	12773
-2	6530	6530	6484	210	1790	4950	6950	6405	6380	12785

The experimental recapitulation on the change in T_{i2} is presented in Table 4. These results indicated that when T_{i2} was increased, the *makespan*, engine completion, transporter completion, Idle M1, Idle M2, Idle M3, Total idle M, Idle T1, Total Idle Transporter increased. Only idle transporter 2 (T2) and Total Idle T decreased. Thus, these results manifested that T_{i2} affected the settlement time.

Table 4. Recapitulation of experiments on changes in T_{i2}

T_{i2}	Makespan	Completion machine	Completion transporter	Idle M1	Idle M2	Idle M3	Total idle M	Idle T1	Idle T2	Total Idle T
5	6535	6535	6489	215	1795	4955	6965	6386	6325	12711
4	6534	6534	6488	214	1794	4954	6962	6385	6336	12721
3	6533	6533	6487	213	1793	4953	6959	6384	6347	12731
2	6532	6532	6486	212	1792	4952	6956	6383	6358	12741
1	6531	6531	6485	211	1791	4951	6953	6382	6369	12751
0	6530	6530	6484	210	1790	4950	6950	6381	6380	12761
-1	6529	6529	6483	209	1789	4949	6947	6380	6391	12771
-2	6528	6528	6482	208	1788	4948	6944	6379	6402	12781
-3	6527	6527	6481	207	1787	4947	6941	6378	6413	12791
-4	6526	6526	6480	206	1786	4946	6938	6425	6376	12801

A recapitulation of the experiment on changes in R_{i2} is presented in Table 5. These results projected that when R_{i2} was increased, the *makespan*, engine completion, transporter completion, Idle M1, Idle M2, Idle M3, Total idle M were the same. Idle transporter 2 (T2) and Total Idle T decreased. On the other hand, idle T1 increased. However, the change in R_{i2} was not significant to the completion time.

Table 5. Recapitulation of calculations on R_{i2}

R_{i2}	makespan	Completion machine	Completion transporter	Idle M1	Idle M2	Idle M3	Total idle M	Idle T1	Idle T2	Total Idle T
3	6530	6530	6487	210	1790	4950	6950	6384	6347	12731
2	6530	6530	6486	210	1790	4950	6950	6383	6358	12741
1	6530	6530	6485	210	1790	4950	6950	6382	6369	12751
0	6530	6530	6484	210	1790	4950	6950	6381	6380	12761
-1	6530	6530	6483	210	1790	4950	6950	6380	6391	12771
-2	6530	6530	6482	210	1790	4950	6950	6379	6402	12781

3.3 Algorithm Comparison

This stage represents the comparison results of the proposed Moth Flame Optimization (MFO) *makespan* algorithm with the heuristic procedure proposed by Hurink and Knust [24]. The results of the comparison of algorithms are presented in Table 6.

Table 6. Comparison of the MFO Algorithm and heuristic procedures

Schedulling Procedure	Job Sequence	Makespan
Hurink and Knust [24]	1-2-3-4-5-6-7-8-9-10-11-12	6,749
Proposed MFO	11-7-1-3-2-6-10-5-12-8-4-9	6,530
Difference		219
Savings Percentage %		3%

These results indicated that the *makespan* MFO produced 6530 minutes in the sequence 11-7-1-3-2-6-10-5-12-8-4-9. Meanwhile, the Hurink and Knust algorithm [24] produced a *makespan* of 6749 minutes in the order of 1-2-3-4-5-6-7-8-9-10-11-12, with a Savings Percentage of 3%. These results indicated that the MFO algorithm was more efficient than the Hurink and Knust [24] algorithm. Furthermore, the proposed MFO algorithm allowed a more comprehensive solution and could reach more local optimum points. Therefore, the opportunity to get a solution that is close to the optimum was even more significant.

4. Conclusion

This study was intended to solve the flow shop scheduling problem with the objective function of minimizing *makespan* considering travel time. The MFO algorithm was proposed to solve this problem. The results showed that the MFO algorithm was able to provide a percentage of savings of 3. The researcher also conducted a sensitivity analysis to test the travel time and return parameters. The results indicated that the smaller the transportation time, the smaller the completion time (*makespan*). This study assumed that job arrival was the same. Therefore, it is also necessary to consider flow shop scheduling with dynamic job arrivals.

Data Availability

All data generated or analyzed during this study are included in this article.

Declarations

Author contribution: All authors contributed equally to the main contributor to this paper. In addition, all authors read and approved the final paper.
Funding statement: No funding was received for this study.
Conflict of interest: The authors declare no conflict of interest.
Additional information: No additional information is available for this paper.

Acknowledgment

We thank the anonymous reviewers for their careful reading of our paper and their many insightful comments and suggestions.

References

[1] K. R. Baker and D. Trietsch, *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.
[2] 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. http://dx.doi.org/10.25077/josi.v18.n2.p133-141.2019.
[3] R. Ginting, "Penjadwalan mesin," *Yogyakarta: Graha Ilmu,* 2009.
[4] D. M. Utama, "Minimizing Number of Tardy Jobs in Flow Shop Scheduling Using A Hybrid Whale Optimization Algorithm," *Journal of Physics: Conference Series,* vol. 1845, no. 1, p. 012017, 2021. http://doi.org/10.1088/1742-6596/1845/1/012017.
[5] D. S. Widodo and D. M. Utama, "The Hybrid Ant Lion Optimization Flow Shop Scheduling Problem for Minimizing Completion Time," *Journal of Physics: Conference Series,* vol. 1569, p. 022097, 2020. http://doi.org/10.1088/1742-6596/1569/2/022097.
[6] D. M. Utama, D. S. Widodo, M. F. Ibrahim, and S. K. Dewi, "An effective hybrid ant lion algorithm to minimize mean tardiness on permutation flow shop scheduling problem," International Journal of Advances in Intelligent Informatics, vol. 6, pp. 23-35, 2020.https://doi.org/10.26555/ijain.v6i1.385.
[7] I. Kuswandi, "Minimasi Maskepan Dengan Penjadwalan Produksi Pada Tipe Produksi Berulang," *Jurnal Teknik Industri,* vol. 11, no. 1, pp. 84-93, 2010. https://doi.org/10.22219/JTIUMM.Vol11.No1.84-93.
[8] A. Sugioko, "Perbandingan Algoritma Bee Colony dengan Algoritma Bee Colony Tabu List dalam Penjadwalan Flow Shop," *Jurnal Metris,* vol. 14, no. 2, pp. 113-120, 2013. http://ojs.atmajaya.ac.id/index.php/metris/article/view/19.
[9] D. M. Utama, A. K. Garside, and W. Wicaksono, "Pengembangan Algoritma Hybrid Flowshop Three-Stage Dengan Mempertimbangkan Waktu Setup," *Jurnal Ilmiah Teknik Industri,* vol. 18, no. 1, pp. 72-78, 2019. https://doi.org/10.23917/jiti.v18i1.7683.
[10] Y. Muharni, E. Febianti, and N. N. Sofa, "Minimasi Makespan Pada Penjadwalan Flow Shop Mesin Paralel Produk Steel Bridge B-60 Menggunakan Metode Longest

Processing Time Dan Particle Swarm Optimization," *Journal Industrial Servicess,* vol. 4, no. 2, pp. 68-75, 2019.doi:http://dx.doi.org/10.36055/jiss.v4i2.5154

[11] M. Hidayat, R. Ekawati, and P. F. Ferdinant, "Minimasi Makespan Penjadwalan Flowshop Menggunakan Metode Algoritma Campbell Dudek Smith (CDS) Dan Metode Algoritma Nawaz Enscore Ham (NEH) Di PT Krakatau Wajatama," *Jurnal Teknik Industri Untirta*, vol. 4, no. 2, 2017. https://jurnal.untirta.ac.id/index.php/jti/article/viewFile/1405.

[12] D. M. Utama, "Analisa perbandingan penggunaan aturan prioritas penjadwalan pada penjadwalan non delay n job 5 machine," in *Prosiding SENTRA (Seminar Teknologi dan Rekayasa)*, 2016, no. 2, pp. 19-23. http://research-report.umm.ac.id/index.php/sentra/article/view/1826.

[13] D. M. Utama, "Algoritma LPT-Branch and Bound Pada Penjadwalan Flexible Flowshop untuk Meminimasi Makespan," *PROZIMA,* vol. 2, no. 1, pp. 20-26, 2019. https://doi.org/10.21070/prozima.v2i1.1527.

[14] D. S. Widodo, P. B. Santoso, and E. Siswanto, "Pendekatan Algoritma Cross Entropy-Genetic Algorithm Untuk Menurunkan Makespan Pada Penjadwalan Flow Shop," *Journal of Engineering Management in Industrial System,* vol. 2, no. 1, pp. 41-49, 2014. https://doi.org/10.21776/ub.jemis.2014.002.01.6

[15] H. A. Shiddiq, S. Sugiono, and C. F. M. Tantrika, "Implementasi Algoritma Simulated Annealing Pada Penjadwalan Produksi Untuk Meminimasi Makespan (Studi Kasus di PT. Gatra Mapan, Karang Ploso, Malang)," *Jurnal rekayasa dan manajemen sistem industri,* vol. 3, no. 1, pp. p43-52, 2015. http://jrmsi.studentjournal.ub.ac.id/index.php/jrmsi/issue/view/17.

[16] H. Wei, S. Li, H. Jiang, J. Hu, and J. Hu, "Hybrid Genetic Simulated Annealing Algorithm for Improved Flow Shop Scheduling with Makespan Criterion," *Applied Sciences,* vol. 8, no. 12, pp. 2621, 2018. http://doi.org/10.3390/app8122621.

[17] F. Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops," *Computers & Industrial Engineering,* vol. 63, no. 2, pp. 355-361, 2012. https://doi.org/10.1016/j.cie.2012.03.015.

[18] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Computers & Operations Research,* vol. 31, no. 11, pp. 1891-1909, 2004. https://doi.org/10.1016/S0305-0548(03)00145-X.

[19] J. E. C. Arroyo and A. A. de Souza Pereira, "A GRASP heuristic for the multi-objective permutation flowshop scheduling problem," *The International Journal of Advanced Manufacturing Technology,* vol. 55, no. 5, pp. 741-753, 2011. http://doi.org/10.1007/s00170-010-3100-x.

[20] S. E. Wiratno, N. Rudi, and B. Santosa, "Algoritma Differential Evolution untuk Penjadwalan Flow Shop Banyak Mesin Dengan Multi Obyektif," *Jurnal Teknik Industri,* vol. 13, no. 1, pp. 1-6, 2012. https://doi.org/10.22219/JTIUMM.Vol13.No1.1-6.

[21] K.-C. Ying, "Minimising makespan for multistage hybrid flowshop scheduling problems with multiprocessor tasks by a hybrid immune algorithm," *European Journal of Industrial Engineering,* vol. 6, no. 2, pp. 199-215, 2012. https://doi.org/10.1504/EJIE.2012.045605.

[22] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," European Journal of Operational Research, vol. 177, pp. 2033-2049, 2007. https://doi.org/10.1016/j.ejor.2005.12.009.

[23] H. F. Rahman, M. N. Janardhanan, L. Poon Chuen, and S. G. Ponnambalam, "Flowshop scheduling with sequence dependent setup times and batch delivery in

supply chain," *Computers & Industrial Engineering,* vol. 158, p. 107378, 2021. https://doi.org/10.1016/j.cie.2021.107378.

[24] J. Hurink and S. Knust, "Makespan minimization for flow-shop problems with transportation times and a single robot," *Discrete Applied Mathematics,* vol. 112, no. 1, pp. 199-216, 2001. https://doi.org/10.1016/S0166-218X(00)00316-4.

[25] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems,* vol. 89, pp. 228-249, 2015. https://doi.org/10.1016/j.knosys.2015.07.006.

[26] H. M. Zawbaa, E. Emary, B. Parv, and M. Sharawi, "Feature selection approach based on moth-flame optimization algorithm," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 4612-4617. https://doi.org/10.1109/CEC.2016.7744378.

[27] T. Lamoudan, F. Elkhoukhi, and J. Boukachour, "Flow shop scheduling problem with transportation times, two-robots and inputs/outputs with limited capacity," *International Journal of Intelligent Computing Research,* vol. 2, no. 1/2/3/, pp. 244-253, 2011.

[28] C.-Y. Lee and V. A. Strusevich, "Two-machine shop scheduling with an uncapacitated interstage transporter," *IIE Transactions,* vol. 37, no. 8, pp. 725-736, 2005. http://doi.org/10.1080/07408170590918290.

[29] C.-L. Li and J. Ou, "Machine scheduling with pickup and delivery," *Naval Research Logistics (NRL),* vol. 52, no. 7, pp. 617-630, 2005. https://doi.org/10.1002/nav.20101.

[30] A. Soukhal, A. Oulamara, and P. Martineau, "Complexity of flow shop scheduling problems with transportation constraints," European Journal of Operational Research, vol. 161, pp. 32-41, 2005. https://doi.org/10.1016/j.ejor.2003.03.002.

[31] D. M. Utama and D. S. Widodo, "An energy-efficient flow shop scheduling using hybrid Harris hawks optimization," *Bulletin of Electrical Engineering Informatics,* vol. 10, no. 3, pp. 1154-1163, 2021. https://doi.org/10.11591/eei.v10i3.2958.