A novel hybrid archimedes optimization algorithm for energy-efficient hybrid flow shop scheduling

Dana Marsetiya Utama [a,1,*], Ayu An Putri Salim [a,2], Dian Setiya Widodo[b,3]

[a] Industrial Engineering Department, University of Muhammadiyah Malang, Malang, Indonesia
[b] Manufacturing Engineering Department, University of 17 Agustus 1945 Surabaya, Indonesia
[1] dana@umm.ac.id; [2] salima@umm.ac.id; [3] dianwidodo@untag.ac.id
* corresponding author

ARTICLE INFO

Article history
Received September 17, 2021
Revised June 27, 2022
Accepted July 18, 2022
Available online July 31, 2022

Keywords
Energy Consumption
Energy-efficient
Hybrid Flow Shop
Scheduling
Archimedes Algorithm

ABSTRACT

The manufacturing sector consumes most of the global energy and had been in focus since the outbreak of the energy crisis. One of the proposed strategies to overcome this problem is to implement appropriate scheduling, such as Hybrid Flow Shop Scheduling. Therefore, this study aims to create a Hybrid Archimedes Optimization Algorithm (HAOA) for solving the Energy-Efficient Hybrid Flow Shop Scheduling Problem (EEHFSP). It is hoped that this helps to provide new insights into advanced HAOA methods for resolving the EEHFSP as the algorithm has the potential to be a more efficient alternative. In this study, three stages of EEHFSP were considered in the problem as well as a sequence-dependent setup and removal times in the second stage. Experiments with three population variations and iterations were presented for testing the effect of HAOA parameters on energy consumption. Furthermore, ten job variations are also presented to evaluate the performance of the HAOA algorithm and the results showed that HAOA iteration and the population did not affect the removal and processing of energy consumption, but impacted that of setup and idle. The comparison of these ten cases revealed that the proposed HAOA produced the best total energy consumption (TEC) when compared to the other algorithms.

1. Introduction

The energy crisis has become one of the most pressing environmental problems facing the world today [1]. The over-exploitation of non-renewable natural resources has led to its depletion and greenhouse effect [2], [3]. It has been discovered that the industrial sector is a major contributor to global energy consumption, with half of the total energy attributed to this sector [4]. Out of the total consumption, the manufacturing industries consumed the most energy. For example, in 2011, the Chinese industrial sector's energy consumption made up 70.82% of the country's total, in which 81.32% was attributed to the manufacturing companies [5]. It has also been found that out of the 31% of electricity consumption the manufacturing companies consumed 90% based on the United States' annual energy consumption in 2012 [6]. This implies that the manufacturing companies urgently need to implement an effective production scheduling [7]–[9] and an efficient energy-saving system [10], in order to reduce energy consumption [11]–[13]. Scheduling and sequencing refer to a resource allocation system that requires efficient management [14], and one of the issues encountered during the process is the Energy-Efficient Hybrid Flow Shop Scheduling Problem (EEHFSP) [15].

EEHFSP has been the subject of many studies that attempted to provide solutions for reducing energy consumption. Several studies have previously discussed this problem under two stages, including Schulz [16] which proposed a Genetic Algorithm (GA), and Du, et al. [17] which introduced Ant Colony Optimization (ACO) algorithm. Tao, et al. [18] also presented the Discrete Imperialist Competitive Algorithm in the two stages of EEHFSP. Furthermore, it has also been performed under three stages such as Zeng, et al. [19] which formulated Particle Swarm Optimization (PSO), and Liu, et al. [20] which utilizeda mixed integer linear programming approach. It is important to note that several other techniques have been proposed to minimize energy consumption, such as shuffled frog-leaping [21], improved genetic [22], and imperialist competitive algorithms [23]. The various recent advanced procedures presented include improved iterated greedy algorithm [24], hybrid collaborative [25], and hybrid salp swarm [26]. These previous studies assumed the setup and removal times include processing time, indicating that the setup and removal time need to be considered when solving EEHFSP.

It has been observed that the EEHFSP was designed for solving problems under 2 and 3 phases, while only a few have examined the sequence-dependent setup time in the aspect of three stages. Therefore, this study aims to develop the EEHFSP using Archimedes Optimization Algorithm (AOA), which is a novel algorithm recently designed for solving optimization problems. Hashim, et al. [27] designed this algorithm by replicating the Archimedes' principles. Several engineering fields that have implemented AOA include design [28], forcasting [29], identification of fuel cell parameter [30], optimization prediction [31], and estimation [32]. Unfortunately, this AOA technique has never been used to solve scheduling problems, especially EEHFSP, thereby motivating scholars to investigate this problem. In this present study, the Hybrid Archimedes Optimization Algorithm (HAOA) is integrated with the Neighborhood Search (NS) procedure to solve the EEHFSP problem by considering the three stages of completing the n-job. The first and third stages were on a single machine while the second stage was on two homogeneous machines. The contributions of this study include (1) improving the special knowledge to overcome three-staged problems in EEHFSP by considering the sequence of dependent set up time, and (2) enriching the procedures for solving EEHFSP problems by proposing a novel HAOA.

2. Methods

2.1 The Proposed HAOA

This section describes the proposed algorithm inspired by the Archimedes' principles, which are integrated in the Neighborhood Search (NS) procedures. Archimedes' principle states that an object immersed in a liquid tends to experience an upward or buoyant force, which is equal to the weight of the liquid displaced by the object. This principle has been implemented by Hashim, et al. [27], in the optimization of AOA. Algorithm 1 presents the AOA pseudo-code, which includes population initiation, population evaluation, and parameter replacement. This AOA was combined with NS procedures in HAOA to improve its performance in each iteration. Furthermore, the performance was improved via NS procedures when searching for solution, as EEHFSP is a combinatorial problem. The Short Rank Value (SRV) principle was also proposed to convert the AOA position into a permutation sequence because the algorithm was designed to solve continuous problems. In this study, EEHFSP was classified as discrete, while the combinatorial was classified as an NP-hard problems. The following discussion outlines the complete steps in the proposed HAOA (Fig. 1).

2.1.1 The initiation of the position and the parameter of the algorithm

The position of the object was initiated with equation (1), in which the $i-th$ object in a N population is denoted as O_i. The vector's upper and lower bounds are represented with ub_i and lb_i. Each object has a vector length denoted as D, which is equal to the total job given. The Archimedes' parameter was initiated by calculating the density initiation (den) and volume (vol) using equation (2), where $rand$ indicates the random number within [0, 1] range. Equation (3) models the acceleration initiation (acc) of the $i-th$ object.

```
Algorithm 1 Pseudo-code Hybrid Archimedes Optimization Algorithm (HAOA) Initialization phase:
Initialize the object's N population
Initialize the object's parameters $t_{max}$, C1, C2, C3, dan C4
Initialize random position, density, and volume objects population using equations (1), (2), and (3).
Apply SRV to convert vector position Aquila to permutation job scheduling
Determine the values for Total Energy Consumption (TEC) in each object
Evaluate initial position and select one best fitness value based on minimal TEC
while ($t \leq t_{max}$) do
    for object i=1,2...,N do
        Updated density and volume based on equation (4)
        Updated transfer and density decreasing factor $TF$ and $d$ using equation (5) and (6)
        if $TF \leq 0.5$ then (exploration phase)
            Updated acceleration using equation (7), normalize acceleration using equation (9), and updated
            position using equation (10)
        else (exploitation phase)
            Updated acceleration using equation (8), normalize acceleration using equation (9), updated
            direction flag F using equation (12), and update position using equation (11)
        end if
    end for
    evaluate each object and select one best fitness value based on minimal TEC
    select $x_{best}, vol_{best}, den_{best}$, and $c_{best}$
    (apply swap rule)
    for $i = 0.1$ x $job\ number$
        Perform swap on the best object
        if (evaluate ($x^{swap}$) < evaluate ($x^{AOA}$))
            $xbest = x^{swap}$
        end if
    end for
    (apply flip rule)
    do a flip operation on on the best object
    for $i = 0.1$ x $job\ number$
        Perform flip on the best object
        if (evaluate ($x^{flip}$) < evaluate ($x^{AOA}$))
            $xbest = x^{flip}$
        end if
    end for
    set $t = t + 1$
end while
return The Optimal solution $x_{best}, vol_{best}, den_{best}$, and $c_{best}$
```

Fig. 1. Pseudo-code Hybrid Archimedes Optimization Algorithm (HAOA) **Initialization phase**

This phase illustrated the evaluation of the object's objective function in the initial population in order to select those with the best fitness values. The objective function of the EEHFSP problems was explained in the method section, specifically the mathematical model sub-section. Furthermore, the SRV principles was proposed for calculating the fitness value of each object by converting the AOA's position to become a permutation sequence. After determining the fitness/objective function values, the x_{best}, vol_{best}, den_{best}, and c_{best} were determined from the initiation stage.

$$O_i = lb_i + rand \times (ub_i - lb_i); \quad i = 1,2, \dots, N \tag{1}$$

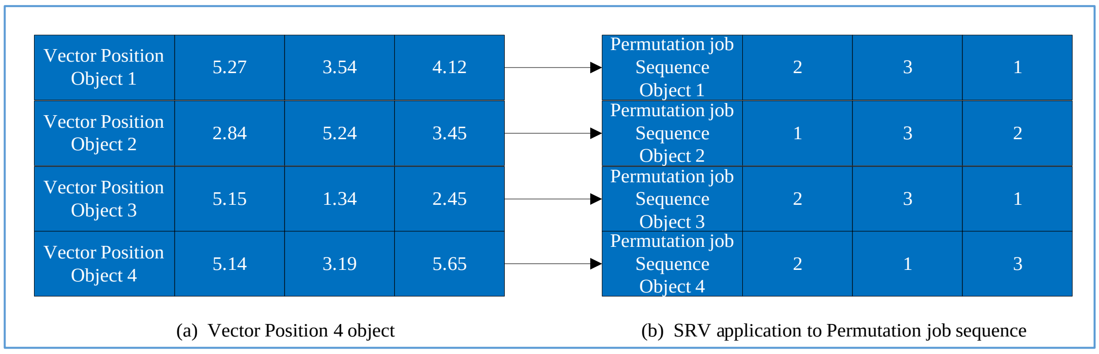
$$\begin{aligned} den_i &= rand \\ vol_i &= rand \end{aligned} \tag{2}$$

$$acc_i = lb_i + rand \times (ub_i + lb_i) \tag{3}$$

2.1.2 The implementation of Short Ranked Value (SRV)

This scheduling for permutation sequence jobs was proposed using the SRV principle. It was observed that the vector length denoted as D in each object was equal to the total number of jobs. The EEHFSP case was divided into three stages, in which the first and third contained the solutions for the job scheduling based on permutation sequence. Afterward, SRV principles were used to solve problems in the first and third stages. The "first come first serve" principle was proposed in the second stage by considering the finishing time in the first stage. Jobs completed in the first stage were prioritized in the second, in which two homogeneous machines were used. Consequently, the machine allocation at the second stage was determined by the lowest load.

It was observed that the SRV principles changed the vector position into a job scheduling sequence to solve EEHFSP problems with the order of values for each object ranging from lowest to highest. Job completion was ranked last in the highest position vector, indicating that the lowest value gets priority for completion. According to Fig. 2a, the vector position of three among the four objects was randomly increased using equation (1) and was converted into job sequence permutation using SRV principles, as shown in Fig. 2. These principles were applied in each iteration when the object's location was changed until the maximum number of iterations was obtained.



Vector Position Object 1	5.27	3.54	4.12	Permutation job Sequence Object 1	2	3	1
Vector Position Object 2	2.84	5.24	3.45	Permutation job Sequence Object 2	1	3	2
Vector Position Object 3	5.15	1.34	2.45	Permutation job Sequence Object 3	2	3	1
Vector Position Object 4	5.14	3.19	5.65	Permutation job Sequence Object 4	2	1	3

(a) Vector Position 4 object (b) SRV application to Permutation job sequence

Fig. 2. The application of SRV in conversion to job scheduling permutation

2.1.3 Densities and volume updates

In the next iteration $(t + 1)$, the volume (den_i^{t+1}) and density (vol_i^{t+1}) parameters of object i are updated based on equation (4), where den_{best} shows the best density in iteration t. vol_{best} was the best volume in the object iteration t. Furthermore, a uniformly distributed random number with a range of $[0,1]$ was denoted by $rand$.

$$\begin{aligned} den_i^{t+1} &= den_i^t + rand \times (den_{best} - den_i^t) \\ vol_i^{t+1} &= vol_i^t + rand \times (vol_{best} - vol_i^t) \end{aligned} \quad (4)$$

2.1.4 Operator transfer and density factor

Initially, the objects collide and after some time, a state of equilibrium was established. This was performed by the AOA algorithm using the transfer operator (TF) value, which changes the search from exploration to exploitation as modeled in equation (5). Furthermore, the transfer value of TF increases during iterations until it reaches 1. The number of iterations and the maximum number of iterations are determined by the variables t and t_{max}.

$$TF = \exp\left(\frac{t - t_{max}}{t_{max}}\right) \quad (5)$$

It was observed that AOA was affected by the factor of density decrease denoted as d, during the global to local search. The decrease of density during the next iteration is modeled by equation (6), in which d^{t+1}decreased as the iteration allowing convergence in the previously selected area increased. It should be pointed out that careful control of this variable ensures a balanced exploration and exploitation in AOA.

$$d^{t+1} = \exp\left(\frac{t_{max} - t}{t_{max}}\right) - \left(\frac{t}{t_{max}}\right) \quad (6)$$

2.1.5 Exploration phase (collision between objects occurs)

In this phase, when $TF \leq 0.5$, the objects are expected to collide. Therefore, the material denoted as mr was selected randomly and the object acceleration in the subsequent iteration (acc_i^{t+1}) was updated based on equation (7). The density, volume, and acceleration of object i on iteration $t + 1$ are denoted as den_i^{t+1}, vol_i^{t+1}, and acc_i^{t+1}respectively. Furthermore, the acceleration, density, and volume of the random object are notated as acc_{mr}, den_{mr}, and vol_{mr}.

$$acc_i^{t+1} = \frac{den_{mr} + vol_{mr} \times acc_{mr}}{den_i^{t+1} \times vol_i^{t+1}} \quad (7)$$

2.1.6 Exploitation phase (no collision between objects)

In this phase, when $TF > 0.5$, a collision between objects does not happen. Therefore, the acceleration formula in the subsequent iteration (acc_i^{t+1}) of the object is shown in equation (8). Where the acceleration object is best, indicated by acc_{best} .

$$acc_i^{t+1} = \frac{den_{best} + vol_{best} \times acc_{best}}{den_i^{t+1} \times vol_i^{t+1}} \quad (8)$$

2.1.7 Normalize acceleration

In this phase, the acceleration of each object in the subsequent iteration (acc_i^{t+1}) was normalized based on equation (9). The normalization range variables are represented by u and l, with each set at 0.9 and 0.1. Furthermore, the acceleration value was set to the highest when object l was far from the global optimum, indicating that it is in the exploration stage, otherwise, the searching progresses to exploitation. This showed that the acceleration factor starts with a large value and decreases as the number of iterations increases.

$$acc_{i-norm}^{t+1} = u \times \frac{acc_i^{t+1} - \min(acc)}{\max(acc) - \min(acc)} + l \tag{9}$$

2.1.8 Update position

The update for the -th object for the subsequent iteration $(t+1)$ uses equation (10) when $TF \leq 0.5$ at the exploration phase. Here, C_1 and C_3 represents the constant which is equal to 2. However, the update for object position employs equation (11) when $TF > 0.5$ at the exploitation phase.

$$x_i^{t+1} = x_i^t + C_1 \times \text{rand} \times acc_{i-norm}^{t+1} \times \text{d} \times (x_{rand} - x_i^t) \tag{10}$$

$$x_i^{t+1} = x_{best}^t + F \times C_2 \times rand \times acc_{i-norm}^{t+1} \times d \times (T \times x_{best} - x_i^t) \tag{11}$$

where C_2 represent the constant variable that is equal to 6. It was observed that the value of the variable increases as the iteration, and it is proportional to the increase in operator transfer TF. The T value is modeled as $C_3 \times TF$ and the value ranges from $[C_3 \times 0{,}3, 1]$. The F value is modeled based on equation (12).

$$F = \begin{cases} +1 & if\ \ P \leq 0.5 \\ -1 & if\ \ P > 0.5 \end{cases} \tag{12}$$
$$\text{where } P = 2 \times rand - C_4$$

2.1.9 Neighborhood Search and evaluation

The two proposed neighborhood search strategies used for improving the performance of the AOA algorithm include flip and swap rules. Fig. 3 shows an illustration of a swap for an object with a 3-dimensional vector, in which two-position vectors are selected at random and then swapped between the selected vectors. The flip technique was also performed by reversing the order of a random selection between 2 position vectors. Fig. 4 depicts an illustration of a flip rule with three vector dimensions, where each iteration t, swap rule, and flip was repeated in $0.1\ x\ job\ number$. It was observed that the SRV procedure converts a new position vector from the neighborhood search each time these two rules are applied. Afterward, the objective function/fitness of the EEHFSP problem was calculated in order to compare it with the previous solution, to observe whether the neighborhood search result solution was possibly better than the one obtained. In this scenario, the fitness value/objective function and the neighborhood search result position vector are stored, but when the solution is not better, the previous solution is retained. For each evaluation, the $xbest$, $volbest$, $denbest$, and $cbest$ values were selected.

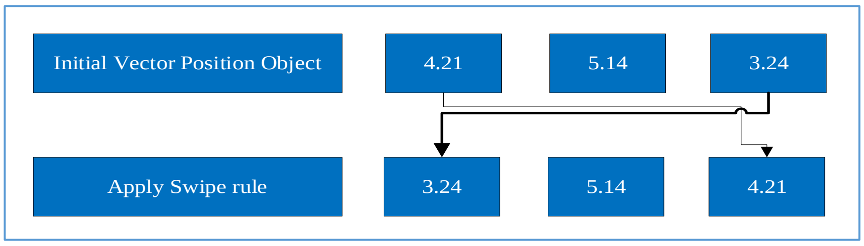


Fig. 3. The illustration of swap in the neighborhood procedure

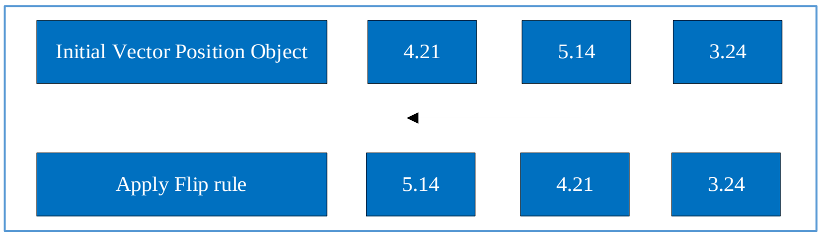


Fig. 4. The illustration of flip in the neighborhood procedure

2.2 Definition of problems, notations, and mathematical models

The EEHFSP was quite difficult to solve in situations where three stages are required to complete n-jobs. In this scenario, stages 1 and 3 are on one machine, while stage 2 involved two identical machines. Fig. 5 shows an illustration of the EEHFSP problem at each stage, in which several assumptions need to be fulfilled, namely (1) This problem consists of three stages with one engine in stages 1 and 3, and two machines in stage 2. (2) Machine 1 and engine 2 in stage 2 have identical characteristics. (3) The initial setup time of each machine is fixed and deterministic. (4) Job setup time in stage 2 is sequence-dependent. (5) At $t = 0$, all machines are available and are shut down after the last job has been completed. (6) Each job was started and completed without interruption. (7) The transfer time was ignored, and (8) Setting time differs from processing time.

To simplify the model, the notations that are used in the mathematical modeling of EEHFSP was employed. The following are the notations used,

Index

i	: index job
k	: index machine
j	: index stage

Variables and parameters

m	: the number of stages
n	: the number of jobs to be completed
TEC	: Total Energy Consumption
EB	: the total energy consumption processing
$EST1$	: the total energy consumption during initial setup
$EST2$	: the total energy consumption setup
M_j	: the number of parallel machines at the j-th stage
P_{ijk}	: processing time of job i at stage j with machine k
N_{jk}	: the number of the job with machine k at the stage j
S_{ijk}	: the starting time of job i at stage j with machine k
ST_{jk}	: the initial set up time of the machine k at stage j
$ST2_{(i,i+1)jk}$	: the setup time of job i and $i + 1$ at stage j with machine k
C_{ijk}	: the completion time of job i at stage j with machine k
R_{ijk}	: the removal time of job i at stage j with machine k
I_{jk}	: idle time stage j with machine k
EI_{jk}	: the energy consumption during idle at stage j with machine k

E_{ijk}	: the energy consumption during the processing of job i at stage j with machine k
ER_{jk}	: the energy consumption during removal at stage j with machine k
$EST1_{jk}$	: the energy consumption during initial setup at stage j with machine k
$EST2_{jk}$	: the energy consumption during setup at stage j with machine k

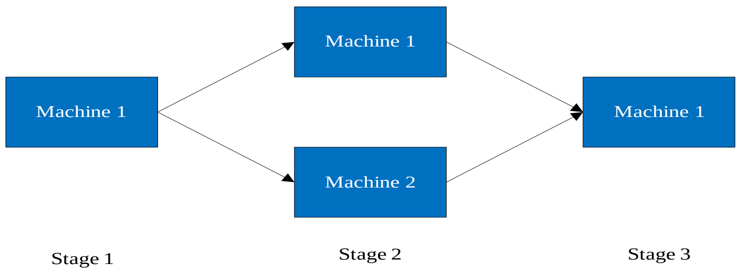


Fig. 5. The illustration of the three-staged EEHFSP

The objective function was to minimize the consumption of energy (TEC). The mathematical model is further elaborated in the following discussion.

$$min\, TEC = EB + EST1 + EST2 + ER + EI \tag{13}$$

Subject to:

$$C_{ijk} \le S_{i(j+1)r} \quad i = 1,2 \dots n, j = 1,2 \dots m, k = 1,2 \dots M_j, r = 1,2 \dots M_{j+1} \tag{14}$$

$$C_{ijk} \le S_{ijk} + ST_{jk} + P_{ijk} + ST2_{(i,i+1)jk} + R_{ijk} \;\; i = 1,2 \dots n, j = 1,2 \dots m, k = 1,2 \dots M_j \tag{15}$$

$$C_{hjk}jk \le S_{(h+1)jk}jk \;\; j = 1,2 \dots m, k = 1,2 \dots M_j, h = 1,2 \dots N_{jk} - 1 \tag{16}$$

$$\sum_{k=1}^{M_j} Y_{ijk} = 1 \qquad i = 1,2,3 \dots n, j = 1,2,3 \dots k \tag{17}$$

$$\sum_{k=1}^{M_j} N_{jk} = n \qquad j = 1,2,3 \dots k \tag{18}$$

$$EB = \sum_{j=1}^{m} \sum_{k=1}^{M_j} \sum_{i=1}^{N_{jk}} E_{ijk} \tag{19}$$

$$EI = \sum_{j=1}^{m} \sum_{k=1}^{M_j} \sum_{i=1}^{N_{jk-1}} EI_{jk}(S_{(h+1)_{jk}jk} - C_{h_{jk}jk}) \tag{20}$$

$$EST1 = \sum_{j=1}^{m} \sum_{k=1}^{M_j} EST1_{jk} \tag{21}$$

$$EST2 = \sum_{j=1}^{m} \sum_{k=1}^{M_j} EST2_{jk} \tag{22}$$

$$ER = \sum_{j=1}^{m} \sum_{k=1}^{M_j} ER_{jk} \tag{23}$$

The formulation of the TEC minimization objective function in the EEHFSP problem is shown in equation (13). Constraint (14) ensures that the job has been processed in the previous stage before being processed at stage $j + 1$. Constraint (15) illustrates that the job completion time in the previous stage is considered, as well as the initial setup time for each job, processing time, and removal time. Furthermore, constraint (16) shows that when the machine completes a job, it is positioned to do the next, and constraint (17) models that each job is only processed on 1 machine in each phase, where variable Y_{ijk} represents a binary number 0 and 1. At each stage of the job to be processed, n is formulated in Constraint (18), and the TEC are modeled in equations (19) to (23).

2.3 Experimental Setting

There were twenty-seven jobs that were completed in the EEHFSP problems, and their processing and removal times were presented in Table 1. Furthermore, the setup times from the i-th to the i + 1 jobs at the second stage is performed, while the initial setup time and energy data at each stage for each time measurement were presented in Table 2. The three experimental levels for each iteration parameter and the population used include 100, 200, and 500. These parameter variations were used to test the energy consumption component to minimize TEC.

The comparison with other variants of the algorithm was also presented and the algorithms used were PSO [19], GA [16], and ACO [17]. A total of 9 variants of problems that were randomly generated were employed based on the uniform distribution range [1.8, 12.3] for the processing time. Furthermore, the removal time was generated following the random integer principle with the range of [1,3], and the setup time data from the i-th to the i + 1 jobs at the second stage were generated from the range of [0,32]. All experiments were conducted using the MATLAB application.

Table 1. Processing and removal time

Job	ProcessingTime			Removal Time		
	stage 1	*stage 2*	*stage 3*	*stage 1*	*stage 2*	*stage 3*
1	7.2	6.5	10	0	2	0
2	6	6.5	8.3	0	2	0
3	7.2	12.3	10	0	1	0
4	7.2	12.3	10	0	1	0
5	3.6	12.3	5	0	1	0
6	7.2	7.3	10	0	1	0
7	3.6	7.3	5	0	1	0
8	7.2	6.3	10	0	2	0
9	7.2	6.3	10	0	2	0
10	7.2	6.1	10	0	2	0
11	7.2	6.1	10	0	2	0
12	2.4	6.1	3.3	0	2	0
13	7.2	12	10	0	3	0
14	7.2	12	10	0	3	0
15	7.2	12	10	0	3	0
16	2.4	12	3.3	0	3	0
17	7.2	5.3	10	0	3	0
18	7.2	5.3	10	0	3	0
19	7.2	9.4	10	0	2	0
20	7.2	9.4	10	0	2	0
21	3.6	9.4	5	0	2	0
22	7.2	7.5	10	0	1	0
23	7.2	7.5	10	0	1	0
24	1.8	7.5	2.5	0	1	0
25	7.2	8.2	10	0	2	0
26	7.2	8.2	10	0	2	0
27	6.6	8.2	9.2	0	2	0

Table 2. Initial setup time and data for energy in each stage for each time measurement

	Stage 1	Stage 2	Stage 3
Intial setup	2	7	10
Energy Consumption Intial setup	0.07	0.4	0.3
Setup Energy Consumption for job i	0.07	0.4	0.3
Processing Energy Consumption	0.25	2	1.8
Removal Energy Consumption	0	0.4	0
Idle Energy Consumption	0.025	0.02	0.18

3. Results and Discussion

3.1 TEC optimization employing HAOA

The results of optimizing energy consumption using the proposed HAOA algorithm are presented in Table 3. It was observed that when the population and iterations used to solve the EEHFSP problem are small, such as 100 populations and iterations respectively, the resulting total energy consumption was huge, and when it was in the medium with about 200 populations and iterations respectively, the total energy consumption produced was better than the one previously performed. Meanwhile, when they are large with about 500 populations and iterations respectively, the resulting total energy consumption was minimal. These results showed that as the population and the iterations applied to HAOA increases, the TEC level becomes low.

Table 3. The results from the optimization of energy consumption with HAOA

Population	Iteration	TEC	EB	EST1+EST2	ER	EI
	100	1,042.28	917.18	92.34	20.80	11.960
100	200	1,039.50	917.18	92.74	20.80	8.784
	500	1,037.92	917.18	91.54	20.80	8.400
	100	1,035.95	917.18	88.74	20.80	9.228
200	200	1,031.38	917.18	84.54	20.80	8.860
	500	1,028.78	917.18	83.14	20.80	7.662
	100	1,029.02	917.18	83.94	20.80	7.096
500	200	1,019.48	917.18	75.94	20.80	5.556
	500	1,016.30	917.18	73.14	20.80	5.182

Furthermore, the experimental results of iteration variations and the HAOA population also show the energy consumption for each parameter, which include processing energy consumption (EB), removal (ER), setup $(EST1 + EST2)$, and idle (EI). It was observed that the value of energy consumption for processing and remedial was constant, indicating that the iteration and population variations do not affect the ER and EB, as they were constant in each variation of the experiment. However, setup and idle energy consumptions decreased as the HAOA iteration and population increased, which indicated that $EST1 + EST2$ and EI have a significant effect on total energy consumption. This is the reason decision makers need to focus on handling setup and idle times in order to minimize total energy consumption.

3.2 Comparison of algorithms

In Table 4, the results of comparing the HAOA algorithm with GA, PSO, and ACO were presented and they were ordered from the smallest job problem, which is20, to the largest being100 jobs. The experiment illustrated that all algorithms produced similar solutions when the jobs were within the range of 20 to 25, but for 27 to 30 jobs, the HAOA and GA algorithms produced better solutions than the PSO and ACO. In the range of 50 to 100, the HAOA produced better TEC solutions compared to PSO, GA, and ACO. It was observed that the HAOA generally showed the best result with the most negligible TEC value when solving 10 variants of the job case.

Table 4. The results of the experiment on parameter variations on the consumption of energy

Job Case	GA	PSO	ACO	HAOA
20	**477.78**	**477.78**	**477.78**	**477.78**
25	**744.85**	**744.85**	**744.85**	**744.85**
27	**1016.3**	1.019,48	1.019,48	**1016.3**
30	**796.48**	815.12	815.79	**796.48**
50	1547.05	1538.93	1542.99	**1531.4**
60	1725.89	1739.43	1732.66	**1725.6**
70	2028.61	2030.03	2029.32	**2025.6**
80	2262.72	2277.5	2270.11	**2253.1**
90	2609.62	2599.2	2604.41	**2568.5**
100	2835.13	2856.13	2845.63	**2830.8**

4. Conclusions

This study proposed the HAOA for solving EEHFSP problems by considering the dependent, setup, and removal times. The solution for EEHFSP were considered under the three stages and the optimization results showed that the HAOA iteration and population do not affect the energy consumption during processing and removal, but has an effect during setup and idle. It was also observed in the comparison with other algorithm variants that HAOA produced more reliable solutions than the PSO, GA, and ACO. Therefore, it is recommended that the HAOA need be developed to solve multi-objective EEHFSP problems in the future.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding statement. None of the authors have received any funding or grants from any institution or funding body for the research.

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] L. Meng, C. Zhang, X. Shao, and Y. Ren, "MILP models for energy-aware flexible job shop scheduling problem," *J. Clean. Prod.*, vol. 210, pp. 710–723, Feb. 2019, doi: 10.1016/j.jclepro.2018.11.021.

[2] L. Shi, G. Guo, and X. Song, "Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment," *Int. J. Prod. Res.*, vol. 59, no. 2, pp. 576–597, Jan. 2021, doi: 10.1080/00207543.2019.1699671.

[3] D. Marsetiya Utama, "An Effective Hybrid Sine Cosine Algorithm to Minimize Carbon Emission on Flow-shop Scheduling Sequence Dependent Setup," *J. Tek. Ind.*, vol. 20, no. 1, pp. 62–72, Feb. 2019, doi: 10.22219/JTIUMM.Vol20.No1.62-72.

[4] H.-X. Qin *et al.*, "An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 69, p. 100992, Mar. 2022, doi: 10.1016/j.swevo.2021.100992.

[5] C. Lu, Q. Liu, B. Zhang, and L. Yin, "A Pareto-based hybrid iterated greedy algorithm for energy-efficient scheduling of distributed hybrid flowshop," *Expert Syst. Appl.*, vol. 204, p. 117555, Oct. 2022, doi: 10.1016/j.eswa.2022.117555.

[6] C. Lu, Y. Huang, L. Meng, L. Gao, B. Zhang, and J. Zhou, "A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers," *Robot. Comput. Integr. Manuf.*, vol. 74, p. 102277, Apr. 2022, doi: 10.1016/j.rcim.2021.102277.

[7] D. M. Utama, "Minimizing Number of Tardy Jobs in Flow Shop Scheduling Using A Hybrid Whale Optimization Algorithm," *J. Phys. Conf. Ser.*, vol. 1845, no. 1, p. 012017, Mar. 2021, doi: 10.1088/1742-6596/1845/1/012017.

[8] D. S. Widodo and D. M. Utama, "The Hybrid Ant Lion Optimization Flow Shop Scheduling Problem for Minimizing Completion Time," *J. Phys. Conf. Ser.*, vol. 1569, no. 2, p. 022097, Jul. 2020, doi: 10.1088/1742-6596/1569/2/022097.

[9] 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," *Int. J. Adv. Intell. Informatics*, vol. 6, no. 1, pp. 23–35, Mar. 2020, doi: 10.26555/ijain.v6i1.385.

[10] J. Chen, L. Wang, and Z. Peng, "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling," *Swarm Evol. Comput.*, vol. 50, p. 100557, Nov. 2019, doi: 10.1016/j.swevo.2019.100557.

[11] D. M. Utama, D. S. Widodo, M. F. Ibrahim, K. Hidayat, T. Baroto, and A. Yurifah, "The hybrid whale optimization algorithm: A new metaheuristic algorithm for energy-efficient on flow shop with dependent sequence setup," *J. Phys. Conf. Ser.*, vol. 1569, no. 2, p. 022094, Jul. 2020, doi: 10.1088/1742-6596/1569/2/022094.

[12] D. M. Utama and D. S. Widodo, "An energy-efficient flow shop scheduling using hybrid Harris hawks optimization," *Bull. Electr. Eng. Informatics*, vol. 10, no. 3, pp. 1154–1163, Jun. 2021, doi: 10.11591/eei.v10i3.2958.

[13] 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," *Int. J. Technol.*, vol. 10, no. 2, pp. 320–331, Apr. 2019, doi: 10.14716/ijtech.v10i2.2194.

[14] Y. Li *et al.*, "A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times," *Int. J. Prod. Res.*, vol. 59, no. 13, pp. 3880–3899, Jul. 2021, doi: 10.1080/00207543.2020.1753897.

[15] Y. Zuo, Z. Fan, T. Zou, and P. Wang, "A Novel Multi-Population Artificial Bee Colony Algorithm for Energy-Efficient Hybrid Flow Shop Scheduling Problem," *Symmetry (Basel).*, vol. 13, no. 12, pp. 1–22, Dec. 2021, doi: 10.3390/sym13122421.

[16] S. Schulz, "A Genetic Algorithm to Solve the Hybrid Flow Shop Scheduling Problem with Subcontracting Options and Energy Cost Consideration," in *Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology*, 2019, pp. 263–273, doi: 10.1007/978-3-319-99993-7_23.

[17] B. Du, H. Chen, G. Q. Huang, and H. D. Yang, "Preference Vector Ant Colony System for Minimising Make-span and Energy Consumption in a Hybrid Flow Shop," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, London: Springer London, 2011, pp. 279–304. doi: 10.1007/978-0-85729-652-8_9

[18] X. Tao, J. Li, T. Huang, and P. Duan, "Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption," *Complex Intell. Syst.*, vol. 7, no. 1, pp. 311–326, Feb. 2021, doi: 10.1007/s40747-020-00193-w.

[19] Zeng Ling-Li, Zou Feng-Xing, Xu Xiao-hong, and Gao Zheng, "Dynamic scheduling of multi-task for hybrid flow-shop based on energy consumption," in *2009 International Conference on Information and Automation*, 2009, pp. 478–482, doi: 10.1109/ICINFA.2009.5204971.

[20] Z. Liu, J. Yan, Q. Cheng, C. Yang, S. Sun, and D. Xue, "The mixed production mode considering continuous and intermittent processing for an energy-efficient hybrid flow shop scheduling," *J. Clean. Prod.*, vol. 246, p. 119071, Feb. 2020, doi: 10.1016/j.jclepro.2019.119071.

[21] D. Lei and T. Wang, "Solving distributed two-stage hybrid flowshop scheduling using a shuffled frog-leaping algorithm with memeplex grouping," *Eng. Optim.*, vol. 52, no. 9, pp. 1461–1474, Sep. 2020, doi: 10.1080/0305215X.2019.1674295.

[22] L. Meng, C. Zhang, X. Shao, Y. Ren, and C. Ren, "Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines," *Int. J. Prod. Res.*, vol. 57, no. 4, pp. 1119–1145, Feb. 2019, doi: 10.1080/00207543.2018.1501166.

[23] M. Li, D. Lei, and J. Cai, "Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives," *Swarm Evol. Comput.*, vol. 49, pp. 34–43, Sep. 2019, doi: 10.1016/j.swevo.2019.05.006.

[24] H.-X. Qin, Y.-Y. Han, Y.-P. Liu, J.-Q. Li, Q.-K. Pan, and Xue-Han, "A collaborative iterative greedy algorithm for the scheduling of distributed heterogeneous hybrid flow shop with blocking constraints," *Expert Syst. Appl.*, vol. 201, p. 117256, Sep. 2022, doi: 10.1016/j.eswa.2022.117256.

[25] J. Mou, P. Duan, L. Gao, X. Liu, and J. Li, "An effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling," *Futur. Gener. Comput. Syst.*, vol. 128, pp. 521–537, Mar. 2022, doi: 10.1016/j.future.2021.10.003.

[26] J. Dong and C. Ye, "Green scheduling of distributed two-stage reentrant hybrid flow shop considering distributed energy resources and energy storage system," *Comput. Ind. Eng.*, vol. 169, p. 108146, Jul. 2022, doi: 10.1016/j.cie.2022.108146.

[27] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems," *Appl. Intell.*, vol. 51, no. 3, pp. 1531–1551, Mar. 2021, doi: 10.1007/s10489-020-01893-z.

[28] B. S. Yıldız, N. Pholdee, S. Bureerat, M. U. Erdaş, A. R. Yıldız, and S. M. Sait, "Comparision of the political optimization algorithm, the Archimedes optimization algorithm and the Levy flight algorithm for design optimization in industry," *Mater. Test.*, vol. 63, no. 4, pp. 356–359, Apr. 2021, doi: 10.1515/mt-2020-0053.

[29] L. Zhang, J. Wang, X. Niu, and Z. Liu, "Ensemble wind speed forecasting with multi-objective Archimedes optimization algorithm and sub-model selection," *Appl. Energy*, vol. 301, p. 117449, Nov. 2021, doi: 10.1016/j.apenergy.2021.117449.

[30] E. H. Houssein, B. E. Helmy, H. Rezk, and A. M. Nassef, "An enhanced Archimedes optimization algorithm based on Local escaping operator and Orthogonal learning for PEM fuel cell parameter identification," *Eng. Appl. Artif. Intell.*, vol. 103, p. 104309, Aug. 2021, doi: 10.1016/j.engappai.2021.104309.

[31] G. Liang, F. Panahi, A. N. Ahmed, M. Ehteram, S. S. Band, and A. Elshafie, "Predicting municipal solid waste using a coupled artificial neural network with archimedes optimisation algorithm and socioeconomic components," *J. Clean. Prod.*, vol. 315, p. 128039, Sep. 2021, doi: 10.1016/j.jclepro.2021.128039.

[32] X. Sun, G. Wang, L. Xu, H. Yuan, and N. Yousefi, "Optimal estimation of the PEM fuel cells applying deep belief network optimized by improved archimedes optimization algorithm," *Energy*, vol. 237, p. 121532, Dec. 2021, doi: 10.1016/j.energy.2021.121532.