Available online at : http://josi.ft.unand.ac.id/

Jurnal Optimasi Sistem Industri

| ISSN (Print) 2088-4842 | ISSN (Online) 2442-8795 |

Research Article

AVOA and ALO Algorithm for Energy-Efficient No-Idle Permutation Flow Shop Scheduling Problem: A Comparison Study

Yolanda Mega Risma *, Dana Marsetiya Utama

Department of Industrial Engineering, Muhammadiyah Malang University, Malang, Indonesia

* Corresponding Author: yolandamegarisma@gmail.com

DOI: 10.25077/josi.v22.n2.p126-141.2023 **Submitted** : January 25, 2023 **Accepted** : November 20, 2023 **Published** : December 18, 2023

ABSTRACT

Global energy consumption is a pressing issue and is predicted to continue increasing between 2010 and 2040. Among the various sectors, the industrial sector, particularly manufacturing, is the main driver of this increase. To effectively address this growing problem and support energy conservation efforts, reducing idle time on production-related machines is critical. The No-Idle Permutation Flow Shop Problem (NIPFSP) and indirectly the need to reduce energy consumption in manufacturing processes are the driving forces behind this study. The African Vultures Optimization Algorithm (AVOA) and the Ant Lion Optimizer (ALO) are two novel metaheuristic algorithms designed to achieve this goal. The effectiveness of both AVOA and ALO was rigorously evaluated across three distinct scenarios: small, medium, and large. Statistical analysis, in the form of independent sample t-tests, was employed to compare the performance of these algorithms. We found that, while both algorithms yielded similar results in the small case, AVOA demonstrated a superior capability in optimizing the NIPFSP in the medium and large cases problem and, consequently, in curbing energy consumption. This implies that AVOA offers a more promising approach to addressing energy consumption concerns in the manufacturing sector, particularly in scenarios involving medium to large-scale production processes. The implementation of such innovative metaheuristic algorithms holds the potential to significantly contribute to global energy conservation efforts while enhancing the efficiency of industrial operations.

Keywords: no-idle permutation flow shop, energy consumption, AVOA, ALO

INTRODUCTION

In recent times, the issue of excessive energy consumption has emerged as a pressing global concern that demands immediate attention [1]. Excessive use of energy leads to problems such as running out of resources and causing the greenhouse effect, which are major factors in climate change [2][3][4]. Specifically, the industrial sector, particularly manufacturing, plays a significant role in high energy use in the world [5][6]. Currently, the majority of manufacturing companies predominantly depend on fossil fuels as their primary source of energy [7], causing companies to be under pressure to use less energy for operations [5]. In general the significant energy consumption in manufacturing primarily occurs when machines are not actively in operation, which is a period referred called idle time [8]. Therefore, it is essential to take steps to minimize or eliminate downtime in machines, greatly reducing total energy consumption. One effective solution being considered is a scheduling strategy that improves how machines are used to reduce energy consumption [9].

The primary aim of reducing energy consumption is to preserve it for future generations and mitigate the impact of energy depletion. This imperative is underscored by previous investigations revealing a dynamic relationship between energy depletion, the development of renewable energy sources, and the reduction of carbon emissions [10]. In 2010, the manufacturing sector of China accounted for a staggering 85% of the nation's total electrical energy usage [11]. Similarly, in 2018, the manufacturing sector of Europe contributed to 77% of the world's energy

consumption [12]. This escalating trend in energy use has persisted since 2010 and is projected to continue for the next 40 years [13]. This trajectory is attributed to the essential role of energy in powering the machines integral to production processes within the manufacturing sector, highlighting the critical need for the effective scheduling of production activities [14].

A pervasive source of energy inefficiency within manufacturing contexts is idle time, a phenomenon prevalent across industries such as glass manufacturing, integrated circuits, ceramics, and fiberglass [15]. The quest for a comprehensive planning approach to address this challenge has engendered a focus on machine scheduling. At the core of this pursuit lies the Permutation Flow Shop Scheduling Problem (PFSP), a complex optimization challenge wherein a set of jobs traverses a predetermined sequence of machines with the objective of minimizing the makespan or total completion time [16]. Within the intricate domain of scheduling complexities, the No-Idle Permutation Flow Shop Scheduling Problem (NIPFSP) emerges as a focal point. NIPFSP accentuates the imperative of eradicating idle time, introducing an additional layer of sophistication to the scheduling paradigm. Under the "no-idle" constraint, each job must progress seamlessly through a series of machines without interruptions, emphasizing the optimization of machine resources and the mitigation of energy waste associated with idle times [15].

The NIPFSP is NP-hard, meaning that finding an optimal solution for large instances of the problem is computationally challenging. Various heuristic and metaheuristic algorithms, mathematical models, and optimization techniques have been developed to address NIPFSP, incorporating techniques such as tabu search [17], differential evolution algorithm [18],[19], variable neighborhood search [20, 21], particle swarm optimization [22], cluster search ┼23], and invasive weed optimization algorithm [24]. Furthermore, alternative approaches, including memetic algorithm-node-edge histogram [25], and bacterial memetic algorithm-simulated annealing [26], have been introduced for the same objective. Beyond addressing tardiness in the NIPFSP problem, various techniques have emerged, such as artificial bee colony [27], discrete water wave optimization [28], and teaching-learning based optimization [29]. Additionally, procedures like differential evolution-genetic algorithm [15] and distribution algorithm cuckoo search [30] have been proposed to minimize tardiness in NIPFSP problems.

As industries grapple with the imperative of balancing production efficiency and energy conservation, the NIPFSP methodologies emerges as a promising avenue. By incorporating the NIPFSP strategy into the operational setting of manufacturing, there exists the potential not only to enhance operational efficiency but also to make significant contributions to the global imperative of sustainable and energy-efficient industrial practices. However, turning off idle machines is not practical because of the complex relationship between energy use, machine efficiency, and the production process [16, 31]. While many results have looked at NIPFSP problem using optimization algorithms, unfortunately, only a limited number of studies have directed their focus toward the energy-efficient aspect in the NIPFSP problem. Among the approaches designed to handle the energy-efficient problem in NIPFSP are the Grey wolf algorithm [32], collaborative optimization algorithm [33], and self-learning of the discrete jaya algorithm [34]. Therefore, understanding that energy efficiency in manufacturing has a big influence on both energy use and the total production process is crucial [35].

In the manufacturing industry, the increasing concern about energy consumption requires innovative solutions. An example of a potential approach to solve this problem includes adjusting planning concepts to decrease energy usage in NIPFSP. Two unique metaheuristic algorithms, namely African Vultures Optimization Algorithm (AVOA) and Ant Lion Optimizer (ALO), are used in the proposed study to significantly reduce energy consumption during NIPFSP. Specifically, AVOA is one of the metaheuristic algorithms that mimic the behavior of African vultures [36]. Furthermore, its effectiveness comprises addressing optimization challenges in different domains, parameter estimation for three-diode solar photovoltaic models [36, 37], solving multi-objective flexible job shop scheduling problems, and optimizing exchange membrane fuel cells [38]. Based on its flexibility, AVOA is recommended as a promising solution, while ALO algorithm is a simulation of the interaction between lion ants and captive ants [39] and has successfully addressed optimization challenges such as 1) planning vehicle routes [40]; and 2) forecasting annual electricity usage [41], and 3) scheduling activities to reduce carbon emissions [42]. However, these algorithms have not been previously used to decrease energy consumption.

There is an important need for investigation using the latest algorithms, particularly focusing on AVOA and ALO. These algorithms are selected in this study and the absence of prior application to specifically tackle NIPFSP to

 DOI: 10.25077/josi.v22.n2.p126-141.2023

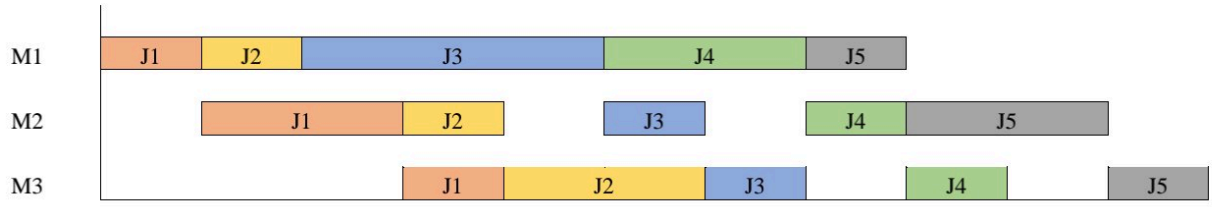


Figure 1. Permutation Flowshop

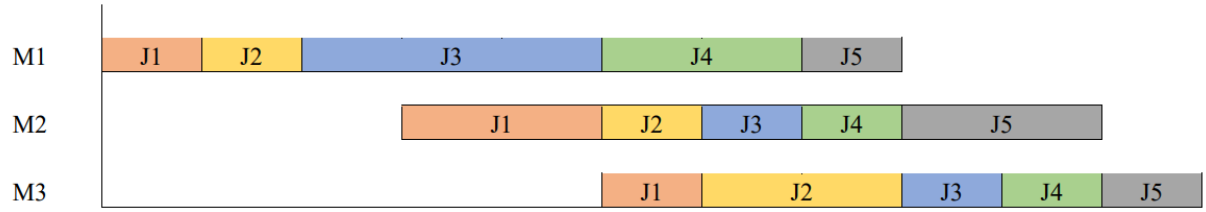


Figure 2 No-Idle Permutation Flowshop

minimize energy consumption. As a result, this research has two main objectives: first, is to develop innovative metaheuristic algorithm that is the AVOA and ALO for solving NIPFSP problem and reducing energy consumption; and second, it compares the outcomes obtained from applying these algorithms to address NIPFSP problems to reduce energy consumption. In the remaining part of this paper, Section 2 provides details on the problem description, example problem, proposed algorithm, and plans for the experimental procedure. Section 3 presents the test results as well as compares the performance of the two algorithms, and finally in Section 4, the research conclusions are presented.

METHODS

Assumptions and Mathematical Formulations

The section provided an overview of the terms and mathematical models related to NIPFSP problem. Figures 1 and 2 showed the Permutation Flow Shop Scheduling Problem (PFSP) and NIPFSP, respectively. In PFSP, the machines were set to stay idle after finishing a task, but in NIPFSP, they were not idle under any circumstances as shown in Figure 2.

In NIPFSP example, several assumptions were made, including:

1. All sets of n jobs had to be processed on m sets of machines in the same order.
2. Jobs arrived and were ready for processing at time 0.
3. The processing start time of the first job on the second to m had to be delayed to follow the no-idle criterion.
4. Only one job could be processed on each machine at a time, and each job could only be processed once.
5. The machine could only stop after the final job had been processed.
6. Job processing time included machine setup time.
7. No machines that were not actively processing jobs were allowed.

Notation:

n	Number of jobs
m	Number of machines
i	Job index
j	Machine index
$C_{i,j}$	The completion time of job i on machine j
S_j	Start time on machine j

F_j	The completion time on machine j
P_{ij}	Processing energy consumption on job i processed in machine j (kWh)
C_{max}	Makespan or completion time (hours)
τ_j	Processing energy consumption on machine j (kWh)
φ_j	Energy consumption when machine j is idle (kWh)
θ_j	Idle time of machine j (hours)
TEC	Total energy consumption (kWh)

Decision Variables:

$$Y_{ijr} = \begin{cases} 1 & \text{if job } i \text{ is processed at speed } r \text{ on machine } j \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ijr} = \begin{cases} 1 & \text{if job } i \text{ is predecessor of job } k \\ 0 & \text{otherwise} \end{cases}, \quad i < k$$

Objective Functions:

Min TEC (1)

Constraints:

$C_{i,1} \geq \sum_{r=1}^{1} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, n)$ (2)

$C_{ij} - C_{ij} - 1 \geq \sum_{r=1}^{1} P_{i1} Y_{ijr} \quad \forall_j = (2, \dots, m),\ i = (2, \dots, n)$ (3)

$C_{ij} - C_{kj} + DX_{ik} \geq \sum_{r=1}^{1} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, n),\ j\ (1, \dots, m),\ k = (1, \dots, n)$ (4)

$C_{ij} - C_{kj} + DX_{ik} \leq D - \sum_{r=1}^{1} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, n),\ j\ (1, \dots, m), k = (1, \dots, n)$ (5)

$C_{max} \geq C_{im} \quad \forall_i = (1, \dots, n)$ (6)

$\sum_{r=1}^{1} Y_{ijr} = 1 \quad \forall i = (1, \dots, n),\ j(1, \dots, m)$ (7)

$Y_{ijr} = Y_{ij+1,r} \quad \forall i = (1, \dots, n),\ j(1, \dots, m),\ r\ (1, \dots, l)$ (8)

$\theta_j = C_{max} - \sum_{i=1}^{n} \sum_{r=1}^{l} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, m)$ (9)

$S_j \leq C_{ij} - \sum_{r=1}^{l} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, n),\ j(1, \dots, m)$ (10)

$F_j \geq C_{ij} \quad \forall i = (1, \dots, n),\ j(1, \dots, m)$ (11)

$F_j \geq S_j + \sum_{i=1}^{n} \sum_{r=1}^{1} P_{i1} Y_{ijr} \quad \forall i = (1, \dots, n),\ j(1, \dots, m)$ (12)

The objective function, as presented in Equation (1) aimed to reduce Total Energy Consumption (TEC). Timestamps for each completed job on the first machine were shown in Equation (2). Furthermore, Equation (3) ensured that the next job could be processed when the previous job had been completed. Job order was detailed in Equation (4) and (5). Meanwhile, the completion time calculation was shown in Equation (6). Equation (7) and (8) confirmed that all jobs were processed at the same speed on every machine. Equation (9) permitted idle time only before and after the final job, there was no idle time on any machine between jobs. Finally, Equation (10), (11), and (12) ensured that each machine remained non-idle while waiting for a job to finish.

Addressing NIPFSP scheduling problem includes determining optimal machine operation time to reduce idle time during job processing. Prior obtaining the optimal operation time, some variables are computed as follow:

$S_j = S_{j-1} + Max_{1 \leq h \leq n} \left\{ \sum_{j=1}^{h} p_{(i),j-1} - \sum_{j=1}^{h-1} p_{(i),j} \right\}$ (13)

$C_{(1),j} = S_j + p_{(1)},\ \forall j = (1,2, \dots, m)$ (14)

$C_{(1),j} = C_{(i-1)j} + p_{(j)} \quad \forall i = (2,3, \dots, n),\ j = (1,2, \dots, m)$ (15)

$C_{max} = Max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{h} C_{(i),j} \right\};\ \forall j = (1,2, \dots, m)$ (16)

$TEC = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{r=1}^{l} \frac{P_{ij} \tau_j}{60} Y_{ijr} + \sum_{j=1}^{m} \frac{\varphi_j \theta_j \tau_j}{60}$ (17)

 DOI: 10.25077/josi.v22.n2.p126-141.2023

Equation (13) shows the formula for machine start time where $S_j = (1,2,\ldots,)$., with the initial value set at $S_1 = 0$. Equation (14) denotes the completion time of the first job on machine j by summing up the starting time of the first job on the first machine and the processing time of the first job on machine j ($p_{(1),j}$). Equation (15) presents the completion time of job i on machine j by summing up the result of the completion time of the previous job on machine j ($C_{(i-1)j}$) with the production process time on machine j ($p_{(j)}$). Equation (16) determines the completion time of the last job on the last machine which is commonly called as makespan. Finally, Equation (17) computes the total energy consumption is presented (TEC).

The Proposed Algorithms

In this section, AVOA and ALO algorithms are introduced for solving the problem of reducing energy consumption in NIPFSP. To achieve the reduction, the Large Ranked Value (LRV) method is used, which is a simple operation known for repositioning the swarm effectively into consecutive permutation jobs and recognized for its efficient mapping of continuous values into permutation jobs (see [43, 44] for more details).

African Vultures Optimization Algorithm (AVOA)

AVOA is a metaheuristic method that exploits the behavior of vultures in their natural habitat. The algorithm uses fitness function calculations to simulate the physical division of vultures into two groups. The strongest and best vultures are the best solutions; the rest of the population strives to move closer to these ideal solutions. To avoid the weakest solutions and improve convergence toward optimal results, the algorithm includes an anti-starvation trade-off strategy. This grouping is inspired by vultures' innate tendency to form groups to effectively find food [36]. The AVOA is further formulated in four stage below:

Stage 1: selection of the best vultures in each group

The populations are initially randomly distributed within the search space. The fitness values of these vultures are then calculated. Among these vultures, the optimal vulture in the first group is selected as the best solution and the superior agent in the second group is picked the second-best solution. The remaining solutions are rearranged into the first and second groups using Equation (18).

$$R(i) = \begin{cases} \text{BestVulture}_1 & \text{if } p_i = L_1 \\ \text{BestVulture}_2 & \text{if } p_i = L_2 \end{cases} \tag{18}$$

where L_1 and L_2 stand for the probability parameter (p_i) that is used to choose the best vulture in the first and second places, respectively. Additionally, the probability that other vultures will move to one of the best solutions is determined by these parameters, which have values between 0 and 1. The roulette wheel mechanism, as shown in the Equation (19), is used to carry out this process.

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \tag{19}$$

where F denotes starvation rate of the vultures.

Stage 2: Starvation rate of vultures

The vultures travel long distances in search of food because they are voracious eaters and have a lot of energy when full. Vultures cannot fly or find food if they are hungry. They also start to act more aggressively. This behavior can be simulated mathematically as follow:

$$t = h\left(sin^w\left(\frac{\pi}{2} \times \frac{iteration_i}{\max\_iteration}\right) + cos\left(\frac{\pi}{2} \times \frac{iteration_i}{\max\_iteration}\right) - 1\right) \tag{20}$$

$$F = (2 \times rand_1 + 1) \times z \times \left(1 - \frac{iteration_i}{maxiterations}\right) + t \tag{21}$$

where F represents the starvation rate of the vultures, *iteration* and *max_iteration* represent the current and maximum iteration, z represents a random number in the interval [-1, 1], h represents a random number in the interval [-2, 2] and $rand_1$ represents a random number in Interval between 0 and 1. The vulture starves when its value is less than 0; However, if its value increases, the vulture is satisfied. The AVOA is also transferred from the exploration to the exploitation phase using Equation (21). Conversely, Equation (20) increases the probability of leaving local optima. Equation (20) uses the term w, a fixed number, to indicate whether the AVOA is in the exploration or exploitation phase. The probability of the exploration phase in the last stage increases as w increases. However, entering the exploration phase is made more difficult by lowering w. In addition, the algorithm enters the exploitation phase if the value of $|F|$ is less than 1, otherwise it enters the exploration phase.

Stage 3: Exploration

The vultures are highly skilled at seeing, finding food, and spotting sick, dying animals. For vultures, however, locating food can be exceedingly challenging. Vultures spend a lot of time examining their living space and cover great distances in pursuit of food. A parameter called P1 is used to select either of the two strategies that vultures can use to investigate different random areas in the AVOA. This parameter, which determines how each of the two strategies is used, needs to be valued before the search operation and should have a value between 0 and 1.

A random number between 0 and 1 is generated to allow the selection of any strategy in the $rand_{p1}$ exploration phase. Equation (23) is applied if this value is greater than or equal to the P1 parameter. On the other hand, Equation (24) is employed if $rand_{p1}$ is less than the parameter P1. In this instance, every vulture in the area looks around it at random to find food. The process is depicted in Equation (22).

$$P(i+1) = \begin{cases} \text{Equation (23)} & \text{if} \quad P_1 \geq rand_{p1} \\ \text{Equation (24)} & \text{if} \quad P_1 < rand_{p1} \end{cases} \tag{22}$$

$$P(i+1) = R(i) - D(i) \times F \tag{23}$$

$$P(i+1) = R(i) - F + rand_2 \times ((ub - lb) \times rand_3 + lb) \tag{24}$$

$$D_{(i)} = |X \times R(i) - P(i)| \tag{25}$$

Equation (23), where $P(i+1)$ is the vulture position vector in the next iteration, states that vultures randomly search the surrounding area at a random distance of one of the best cultures of the two groups. Equation (18) is used in this iteration to select one of the best vultures $R(i)$, as shown in Equation (25). Additionally, vultures move randomly in X to defend food from other vultures. The coefficient vector X (i.e., the value equals $2 \times rand$) which varies with each iteration, increases the random motion (*rand* is a random number [0, 1]).

A random value [0, 1] is used for $rand_2$ in Equation (24). The values of lb and ub show the variables' upper and lower bounds, respectively. Equation (24) is used to generate a basic model for the random generation of solutions in the interval (lb, ub). The coefficient of randomness is increased by using $rand_3$. A random motion is added to the lb if $rand_3$ is given a value that is close to 1. This causes the solutions to be distributed in similar. In order to look for various search space areas and boost diversity, it generates a high random coefficient at the search environment scale.

Stage 4: Exploitation

The fourth phase was the Exploitation phase, occurring when the |F| value ranged between 1 and 0.5. The parameter P_2, which ranged from 0 to 1, determined the selection of each strategy, guided by the random number $rand_{P2}$ and the explanation of this process was provided in Equation (26).

$$P(i+1) = \begin{cases} \text{Equation (27)} & \text{if} \quad P_2 \geq rand_{p2} \\ \text{Equation (30)} & \text{if} \quad P_2 < rand_{p2} \end{cases} \tag{26}$$

In the context of food competition, when the $|F|$ value exceeded 0.5, vultures were relatively well-fed and energetic. The value of $D(i)$ was calculated using Equation (27), where the satiation level of vultures was represented by F. The

 DOI: 10.25077/josi.v22.n2.p126-141.2023

random number $rand_4$, ranging from 0 to 1, and $d(t)$, representing the distance of weaker vultures to one of the two best vulture groups, were calculated through Equation (28). The values for S_1 and S_2, derived from Equation (29), showed vultures engaging in rotational flight using the Spiral model mathematically. The value of $rand_5$ and $rand_6$ represented random numbers in the range of (0,1).

$$P(i + 1) = D(i) \times (F + rand_4) - d(t) \quad (27)$$

$$d(t) = R(i) - P(i) \quad (28)$$

$$S_1 = \mathrm{R(i)} \times \left(\frac{rand_5 \times P(i)}{2\pi}\right) \times \cos(\mathrm{P(i)}) \quad (29)$$

$$S_2 = \mathrm{R(i)} \times \left(\frac{rand_6 \times P(i)}{2\pi}\right) \times \sin(\mathrm{P(i)})$$

$$P(i + 1) = R(i) - (S_1 + S_2) \quad (30)$$

Stage 5: Exploitation (Second phase)

When the value of F was less than 0.5, the part of the process was executed. In the second phase, the random number $rand_{P3}$ was generated in the range of 0 to 1. When $rand_{P3}$ was higher than or equal to P_3 the selected strategy included collecting different types of vultures in the vicinity of the food source. However, when the obtained value was lower than the parameter P_3, the strategy used was encirclement and aggressive competition for food, as explained in Equation (31).

$$P(i+1) = \begin{cases} \text{Equation (33)} & \text{if} \quad P_3 \geq rand_{p3} \\ \text{Equation (34)} & \text{if} \quad P_3 < rand_{p3} \end{cases} \quad (31)$$

To express the movement of vultures, Equation (32) was used. The variable $BestVulture_1$(i) represented the best vulture of the first group, while $BestVulture_2$(i) represented the best vulture of the second group in the current computation. In the current iteration, the satiation level of the vultures represented by the letter F was determined using Equation (21). Furthermore, the position of the i-th vulture was represented by the symbol $P(i)$. The position of the vulture was improved in Equation (33), where $P(i+1)$ represented the position of the vulture in the subsequent iteration, and A_1 and A_2 were derived from Equation (32).

$$A_1 = BestVulture_1(\mathrm{i}) - \frac{BestVulture_1(i) \times P(i)}{BestVulture_1(i) \times P(i)^2} \times \mathrm{F} \quad (32)$$

$$A_2 = BestVulture_2(\mathrm{i}) - \frac{BestVulture_2(i) \times P(i)}{BestVulture_2(i) \times P(i)^2} \times \mathrm{F}$$

$$P(\mathrm{i} + 1) = \frac{A_1 + A_2}{2} \quad (33)$$

In situations of high competition for food, shown by an F value less than 0.5, vulnerable vultures moved toward superior vultures to search for any remaining food pieces. This movement was mathematically represented by Equation (34). Following that, a Lévy flight (LF) pattern was used to improve the effectiveness of the AVOA mechanism. The variable d means the dimension of the problem, and the variables u and v were random numbers in the range of 0 to 1. Equation (35) ensured that the constant value β was equal to 1.5.

$$P(i + 1) = R(i) - |d(t)| \times F \times Levy(d) \quad (34)$$

$$LF(\mathrm{x}) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \quad \sigma = \left(\frac{r(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{r(1+\beta 2) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}}\right)^{\frac{1}{\beta}} \quad (35)$$

This research employs AVOA to address the challenge of reducing energy consumption in the context of NIPFSP. The AVOA pseudocode for the NIPFSP incorporating energy consumption reduction is shown in Figure 3.

```
: Inputs : the population size N and maximum number of iterations T
: Outputs : The location of Vulture and its fitness value
: Initialize the random population P_i (i = 1,2,…,N)
: while (stopping condition is not met) do
:    Apply LRV for changes best position to job sequences
:    Energy Consumption values of Vulture
:    Set P_BestVulture1, as the location of Vulture (First best location Best Vulture Category 1)
:    Set P_BestVulture2, as the location of Vulture (Second best location Best Vulture Category 2)
:    for (each Vulture (P_i)) do
:    Select R(i) using Eq. (18)
:    Update the F using Eq. (21)
:    if (|F| ≥ 1) then
:        if (P_1 ≥ rand_P1) then
:            Update the location of Vulture using Eq. (23)
:        else
:            Update the location of Vulture using Eq. (24)
:    if (|F| < 1) then
:        if (|F| >0.5) then
:            if (P_2 ≥ rand_P2) then
:              Update the location Vulture using Eq. (27)
:          else
:              Update the location Vulture using Eq. (30)
:        else
:            if (P_3 ≥ rand_P3) then
:              Update the location Vulture using Eq. (33)
:          else
:              Update the location Vulture using Eq. (34)
: Return P_BestVulture1
```

Figure 3. AVOA pseudocode for NIPFSP with energy consumption reduction

Ant Lion Optimizer (ALO)

The ALO is an algorithm that mimics the behavior of the Antlion species in hunting prey. In this case, ants will be used as prey for Antlion. Antlion will initially be in a place and make a sand pit trap. Ants that are on the edge of the sand pit looking for food will slip into the hole and become prey for the Antlion. Then the Antlion will move and set a trap to catch other ants. The ALO steps are described as follow:

Step 1: Random walk of ants

The step size t is represented by the random walk of ants, $X(t)$, where n is the maximum number of iterations and csum is the cumulative sum of consecutive random jumps, as expressed in Equation (36). Equation (37) represents the stochastic function $r(t)$ using *rand*, a uniform distribution random number generated within the interval [0, 1].

$$X(t) = [0, cumsum\,(2r(t_1) - 1), cumsum\,(2r(t_2) - 1) \ldots, cumsum(2r(t_n) - 1)] \tag{36}$$

$$r(t) = \begin{cases} 1 & \text{if } rand > 0.5 \\ 0 & \text{if } rand \leq 0.5 \end{cases} \tag{37}$$

 DOI: 10.25077/josi.v22.n2.p126-141.2023

An objective function determines the fitness values for the ants, which are kept in the form of a matrix. M_{Ant} is a matrix for tracking each ant's location, and $A_{i,j}$ represents j-th variable dimension of the i-th ant. Equation (38) defined this formulation, where n denoted the number of ants and d the number of variables. Equation (39) stores the objective function f which represents the fitness values for the ants. The antlions' fitness values are kept in a different matrix in a similar fashion, as expressed in Equation (40) and (41).

$$M_{Ant} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & \cdots & A_{1,d} \\ A_{2,1} & A_{2,2} & \cdots & \cdots & A_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n,1} & A_{n,2} & \cdots & \cdots & A_{n,d} \end{bmatrix} \tag{38}$$

$$M_{OA} = \begin{bmatrix} f([A_{1,1} & A_{1,2} & \ldots & A_{1,d}]) \\ f([A_{2,1} & A_{2,2} & \ldots & A_{2,d}]) \\ & \vdots & & \\ & \vdots & & \\ f([A_{n,1} & A_{n,2} & \ldots & A_{n,d}]) \end{bmatrix} \tag{39}$$

$$M_{Antl} = \begin{bmatrix} AL_{1,1} & AL_{1,2} & \cdots & \cdots & AL_{1,d} \\ AL_{2,1} & AL_{2,2} & \cdots & \cdots & AL_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{n,1} & AL_{n,2} & \cdots & \cdots & AL_{n,d} \end{bmatrix} \tag{40}$$

$$M_{OAL} = \begin{bmatrix} f([AL_{1,1,} & AL_{1,2,} & \ldots & AL_{1,d}]) \\ f([AL_{2,1,} & AL_{2,2,} & \ldots & AL_{2,d}]) \\ & \vdots & & \\ & \vdots & & \\ f([AL_{n,1,} & AL_{n,2,} & \ldots & AL_{n,d}]) \end{bmatrix} \tag{41}$$

Equation (42) controlled the random walk of ants, constraining movement in the search space. This formula used a normalized min-max calculation, where a_i denotes the minimum of the random walk for the i-th variable, c_i represents the minimum value for that iteration, d_i represents the maximum of the random walk for the i-th variable, c_i^t signifies the minimum value for the i-th iteration, and d_i^t is the maximum value for the i-th variable at the t-th iteration.

$$X_i^t = \frac{(X_i^t - a_i)x\,(d_i - c_i^t)}{(d_i^t - a_i)} + c_i \tag{42}$$

Step 2: Falling ants toward ant lion

Random ant movement, affected by ant-lion traps, was expressed through Equations (43) and (44). In Equations (43) and (44), $Antlion_j^t$ represens the position of selecting the j-th ant lion in t-th iteration. c^t represents the minimum value across all variables in t-th iteration, d^t represents a vector with the maximum values across all variables in iteration t, c_i^t represents for the minimum values across all variables for the i-th ant, and d_i^t shows the maximum values across all variables for the i-th ant.

$$c_i^t = Antlion_j^t + c^t \tag{43}$$

$$d_i^t = Antlion_j^t + d^t \tag{44}$$

The narrowing of the random walk limit is shown in Equations (45) and (46), where I is the ratio defined as $I = 10^w \frac{t}{T}$. The notation t represents the current iteration, T is the maximum number of iterations, and w is a constant

based on the current iteration (w = 2 for t > 0.1T; w = 3 for t > 0.5T; w = 4 for t > 0.75T; w = 5 for t > 0.9T; and w = 6 for t > 0.95T), The w value adjusts the exploitation accuracy rate.

$$c^t = \frac{c^t}{I} \tag{45}$$

$$d^t = \frac{d^t}{I} \tag{46}$$

Equation (47) shows how an ant lion captures its prey and then reconstructs the burrow of the ant to bury them in the sand. In this equation, t represents the current repetition, and $antlion_j^t$ shows the position of selecting the j-th ant lion during the t-iteration. Equation (48) models the process of determining the most suitable ant lion. ant_i^t represents the position of the i-th ant during the t-iteration. Using the roulette wheel at the t-iteration, R_A^t describes the position of the ant around the selected ant lion, while R_E^t shows the location where the ant circulated around the selectivity, the best ant lion, during the t-iteration.

$$Antlion_j^t = Ant_i^t \; if \; f(Ant_i^t) > f(Antlion_j^t) \tag{47}$$

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{48}$$

The adopted ALO pseudocode for the NIPFSP integrating energy consumption reduction is presented in Figure 4:

```
1  : Initialize the first population of ants and ant-lions randomly
2  : Apply LRV for changes best position to job sequences
3  : Energy Consumption of ants and ant-lions
4  : Find the best ant lions and assume it as the elite
5  : while (the stopping criterion is not satisfied) do
6  :     for every ant
7  :         Select an ant-lions using Roulette wheel
8  :         Update c and d using equations Eqs. (45) and (46)
9  :         Create a random walk and normalize it using Eqs.(36) and (42)
10 :         Update the position of ant using (48)
11 :     end for
12 :     Calculate the fitness of all ants
13 :     Replace an ant-lion with its corresponding ant it if becomes fitter (Eq.47))
14 :     Update elite if an ant-lion becomes fitter than the elite
15 : end while
16 : return elite
```

Figure 4. ALO pseudocode for NIPFSP with energy consumption reduction

Experimental Data and Procedures

This study used job and machine data from previous study of [45, 46]. Table 1 organized this data to clarify processing times into three case types. Tables 2, 3, and 4 provided details on energy consumption during processing and idle time for Case 1, Case 2, and Case 3, respectively. The experimental procedure included repeating 30 times for both algorithms to establish the total energy consumption (TEC). Each case applied three iterations (100, 300,500), and three populations (100, 300,500) to find the optimal energy consumption. The best performance of both algorithms

 DOI: 10.25077/josi.v22.n2.p126-141.2023

was determined based on the lowest average *TEC* value. IBM SPSS Statistics 21 was used to conduct an independent sample t-test to test the two-tailed significance value and determine the superiority between the AVOA and ALO algorithms. The algorithms' performance equivalents for AVOA and ALO were taken into consideration when the two-tailed significance value was greater than 0.05. On the other hand, AVOA and ALO demonstrated notable performance differences when the value was less than 0.05. The experimental results were also presented through box and whisker diagrams. All these experiments were performed on the Windows 10 operating system using an Intel Core i3 processor.

Table 1. Source of Experimental Data

Problem	Job and Machine	Category	Reference
Case 1	10 job 6 machine	Small	Carlier [45]
Case 2	30 job 10 machine	Medium	Reeves [46]
Case 3	50 job 10 machine	Large	Reeves [46]

Table 2. Energy Consumption Data for Case 1

Job	Machine					
	1	2	3	4	5	6
τ_j (kWh)	0,4507	0,5582	0,3014	0,9409	0,7859	0,2709
φ_j (kWh)	0,0066	0,0164	0,0048	0,1007	0,0871	0,003

Table 3. Energy Consumption Data for Case 2

Job	Machine									
	1	2	3	4	5	6	7	8	9	10
τ_j (kWh)	0.2069	0.4754	0.9815	0.3853	0.1825	0.5439	0.2389	0.5302	0.6656	0.794
φ_j (kWh)	0.00624	0.0096	0.0236	0.0079	0.0048	0.007	0.0033	0.0089	0.008	0.00751

Table 4. Energy Consumption Data for Case 3

Job	Machine									
	1	2	3	4	5	6	7	8	9	10
τ_j (kWh)	0.8785	0.2544	0.884	0.1757	0.6843	0.4324	0.4011	0.1903	0.4123	0.3214
φ_j (kWh)	0.0091	0.002	0.038	0.0009	0.0048	0.0034	0.0033	0.0016	0.0038	0.0041

RESULTS AND DISCUSSION

The results section analyzed the research findings, exploring how the number of repetitions and population size affected TEC. Additionally, it compared the performance usefulness between AVOA and ALO algorithms through an independent sample *t*-test. The subsequent sections provided a detailed discussion of the results for both AVOA and ALO algorithms. The TEC results for various populations and iterations from ALOA and ALO were displayed in Table 4. In Case 1, iteration 300 produced the best results with a population of 100. The optimal performance was reached in Case 2 at iteration 500 with a population of 500, and in Case 3 at the same iteration and population of 500. Increased population and iteration sizes were found to reduce TEC, according to the experimental results. Conversely, lower TEC was produced by smaller populations and fewer repetitions.

To determine the usefulness of the two algorithms, the independent sample *t*-test was performed on all TEC data derived from 30 replicates of experiments for each case and algorithm. Results are displayed using box and whisker plots. The average of the figures shows that for case 1, the data distribution produced by the AVOA algorithm is larger

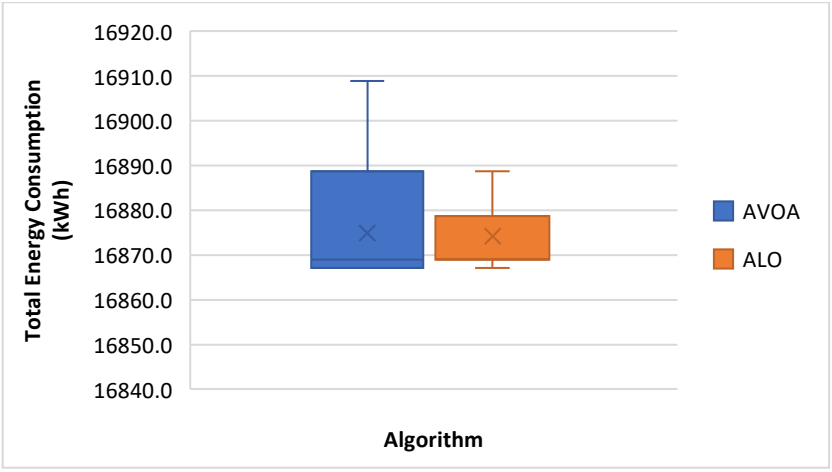
Table 4. Comparison of ALOA and ALO on TEC (in kWh)

Population	Iteration 100		Iteration 300		Iteration 500	
	AVOA	ALO	AVOA	ALO	AVOA	ALO
Case 1						
100	16,867.14	16,868.96	16,867.14	16,867.134	16,867.14	16,867.14
300	16,867.14	16,867.14	16,867.14	16,867.134	16,867.14	16,867.14
500	16,867.14	16,867.14	16,867.14	16,867.134	16,867.14	16,867.14
Case 2						
100	7,612.51	7,615.49	7,613.62	7,613.37	7,612.314	7,613.39
300	7,612.77	7,614.33	7,612.83	7,614.35	7,611.67	7,613.56
500	7,612.52	7,614.50	7,611.99	7,613.59	7,611.67	7,613.82
Case 3						
100	11,993.39	11,993.40	11,993.22	11,992.97	11,993.19	11,994.11
300	11,993.15	11,994.49	11,992.89	11,992.85	11,992.37	11,992.90
500	11,992.84	11,993.94	11,992.57	11,992.82	11,991.92	11,992.36

than the ALO data distribution; However, for cases 2 and 3, as shown in Figures 5b and 5c, respectively, the data distribution produced by the AVOA algorithm is smaller than the ALO data distribution.

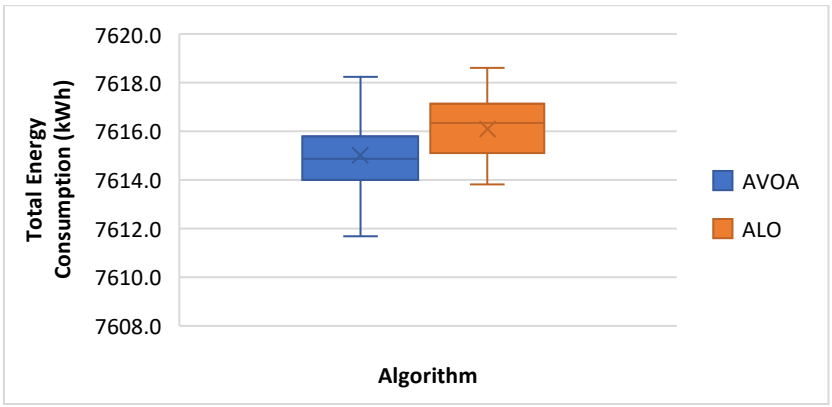
The independent sample t-test results for each case using the AVOA and ALO algorithms are displayed in Table 5. Since the two-tailed significance (sig) value in Case 1 is greater than 0.05, the performance of the AVOA and ALO algorithms is comparable. The AVOA and ALO algorithms, however, perform differently in Cases 2 and 3, with sig values (2-tailed) < 0.05. As a result, the average of the experimental results is used to determine the best performance in Cases 2 and 3. In Cases 2 and 3, the average values derived from AVOA algorithm are lower than ALO's, indicating that AVOA performs better in these scenarios.

As a metaheuristic optimization technique, AVOA sought to mimic the foraging and navigational habits of African vultures [46]. Moreover, earlier studies demonstrated that when AVOA was used in place of alternative metaheuristic algorithms, the results were consistently better [36]. Achieving varying results required a successful trade-off between exploration and exploitation. One important factor in AVOA's effectiveness was how well it handled this trade-off [47]. A time-varying mechanism and chaotic tent mapping were added to an improved version of AVOA, which performed better than other metaheuristic algorithms [48]. AVOA's efficacy was greatly enhanced by its capacity to replicate the African vulture's searching and navigational patterns, which enabled it to explore and utilize the search space more efficiently. As a result, the study suggested solving NIPFSP with AVOA in order to lower energy consumption.

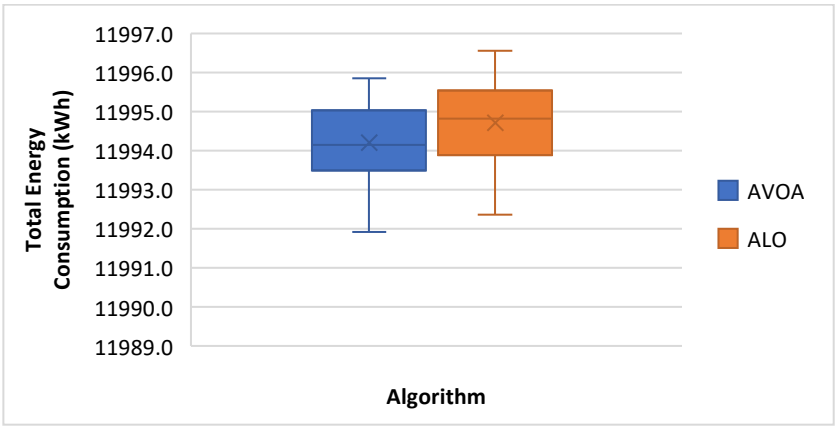


a. Data distribution for Case 1

 DOI: 10.25077/josi.v22.n2.p126-141.2023



b. Data distribution for Case 2



c. Data distribution for Case 3

Figure 5. The independent sample *t*-test for TEC

Table 5. Independent Sample T-test Results

	Case 1		Case 2		Case 3	
	AVOA	ALO	AVOA	ALO	AVOA	ALO
Mean	16,874.938	16,874.22	7,615.021	7,616.097	11,994.204	11,994.718
Standard Deviation	12.176	8.257	1.531	1.302	0.979	0.987
t	0.266		-2.933		-2.025	
Sig.(2-tailed)	0.791		0.005		0.047	

CONCLUSION

This research introduced two innovative approaches, AVOA and ALO, as effective methods to reduce energy consumption. The investigation compared both algorithms to address NIPFSP problem and the results showed that both AVOA and ALO effectively decreases energy consumption in a small dataset. However, in medium and large datasets, AVOA algorithm showed greater effectiveness in minimizing energy consumption. Tests on the parameters of AVOA and ALO showed that energy consumption decreases with larger population sizes and more replication. It should be acknowledged that this exploration has some limitations, particularly in the aspect of computational effort. Therefore, future research is needed to investigate the computational time differences between AVOA and ALO algorithms.

ACKNOWLEDGEMENT

The author would like to sincerely thank the editor and reviewers for their constructive feedback and thorough evaluation of the manuscript. Your insightful comments and recommendations have contributed significantly to improving the overall quality of this work. The author would also like to thank the optimization laboratory of the Department of Industrial Engineering. The use of the laboratory's resources and facilities has played a critical role in the successful completion of this research.

CONFLICT OF INTEREST

This article is presented with the utmost integrity, as the authors attest to the lack of any conflicts of interest that could undermine the veracity or objectivity of the research within the academic domain.

FUNDING

The authors received no financial support for the research, authorship, and/or publication of this article.

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," A Rep. by Anal. Press. Complet. Req. New York Times, vol. 9, no. 2011, p. 161, 2011.

[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, 2021.

[3] D. M. 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, 2019.

[4] Y. Xu and F. Zhao, "Impact of energy depletion, human development, and income distribution on natural resource sustainability," Resour. Policy, vol. 83, p. 103531, 2023, doi: https://doi.org/10.1016/j.resourpol.2023.103531.

[5] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," J. Manuf. Syst., vol. 30, no. 4, pp. 234–240, 2011.

[6] D. M. Utama, A. A. P. Salim, and D. S. Widodo, "A novel hybrid archimedes optimization algorithm for energy-efficient hybrid flow shop scheduling.," Int. J. Adv. Intell. Informatics, vol. 8, no. 2, 2022.

[7] X. Wu and A. Che, "A memetic differential evolution algorithm for energy-efficient parallel machine scheduling," Omega, vol. 82, pp. 155–165, 2019.

[8] G. Mouzon, M. B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," Int. J. Prod. Res., vol. 45, no. 18–19, pp. 4247–4271, 2007.

[9] I. Surjandari, A. Rachman, D. A. Purdianta, and A. Dhini, "The Batch Sheduling Model for Dynamic multiitem, Multilevel Production in an assembly Job-Shop with Parrallel Machines.," Int. J. Technol., vol. 1, pp. 84–96, 2015.

[10] M. R. Hossain, S. Singh, G. D. Sharma, S.-A. Apostu, and P. Bansal, "Overcoming the shock of energy depletion for energy policy? Tracing the missing link between energy depletion, renewable energy development and decarbonization in the USA," Energy Policy, vol. 174, p. 113469, 2023, doi: https://doi.org/10.1016/j.enpol.2023.113469.

[11] Y. Zhao et al., "A comparative study of energy consumption and efficiency of Japanese and Chinese manufacturing industry," Energy Policy, vol. 70, pp. 45–56, 2014.

[12] J. Walther and M. Weigold, "A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry," Energies, vol. 14, no. 4, p. 968, 2021.

[13] C. Clauser and M. Ewert, "The renewables cost challenge: Levelized cost of geothermal electric energy compared to other sources of primary energy–Review and case study," Renew. Sustain. Energy Rev., vol. 82, pp. 3683–3693, 2018.

[14] D. M. Utama, "Pengembangan algoritma neh dan cds untuk meminimasi consumption energy pada penjadwalan flow shop," in Prosiding SENTRA (Seminar Teknologi dan Rekayasa), 2019, no. 4, pp. 47–54.
[15] P. J. Kalczynski and J. Kamburowski, "A heuristic for minimizing the makespan in no-idle permutation flow shops," Comput. Ind. Eng., vol. 49, no. 1, pp. 146–154, 2005.
[16] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and T. Jin Chua, "A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion," Int. J. Prod. Res., vol. 49, no. 16, pp. 5033–5050, 2011.
[17] W.-J. Ren et al., "Tabu search algorithm for solving No-idle permutation Flow Shop Scheduling Problem," in 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA). IEEE, 2010.
[18] Q.-K. Pan and L. Wang, "A novel differential evolution algorithm for no-idle permutation flow-shop scheduling problems," European Journal of Industrial Engineering, vol. 2, no. 3, pp. 279-297, 2008.
[19] G. Deng and X. Gu, "A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion," Computers Operations Research, vol. 39, no. 9, pp. 2152-2160, 2012.
[20] L. Shen et al., "A General Variable Neighborhood Search for the NoIdle Flowshop Scheduling Problem with Makespan Criterion," in 2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE.
[21] H. Öztop et al., "A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling," in 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE.
[22] Q.-K. Pan and L. Wang, "No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm," The International Journal of Advanced Manufacturing Technology, vol. 39, no. 7, pp. 796-807, 2008.
[23] M. S. Nagano and J. C. S. S. Januário, "Evolutionary heuristic for makespan minimization in no-idle flow shop production systems," Acta Scientiarum. Technology, vol. 35, no. 2, pp. 271-278, 2013.
[24] F. Zhao, L. Zhang, J. Cao, and J. Tang, "A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem," Comput. Ind. Eng., vol. 153, p. 107082, 2021.
[25] W. Shao, D. Pi, and Z. Shao, "Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion," Applied Soft Computing, vol. 54, pp. 164-182, 2017.
[26] A. Agárdi et al., "A Hybrid Discrete Bacterial Memetic Algorithm with Simulated Annealing for Optimization of the Flow Shop Scheduling Problem," Symmetry, vol. 13, no. 7, p. 1131, 2021.
[27] M. F. Tasgetiren et al., "A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion," Applied Mathematical Modelling, vol. 37, no. 10-11, pp. 6758-6779, 2013.
[28] F. Zhao et al., "A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion," Expert Systems with Applications, vol. 146, p. 113166, 2020.
[29] 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, no. 4, pp. 523-529, 2017.
[30] Z. Sun and X. Gu, "Hybrid algorithm based on an estimation of distribution algorithm and cuckoo search for the no idle permutation flow shop scheduling problem with the total tardiness criterion minimization," Sustainability, vol. 9, no. 6, p. 953, 2017.
[31] 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," in Journal of Physics: Conference Series, 2020, vol. 1569, no. 2, p. 22094.
[32] C. N. Al-Imron, D. M. Utama, and S. K. Dewi, "An Energy-Efficient No Idle Permutations Flow Shop Scheduling Problem Using Grey Wolf Optimizer Algorithm," J. Ilm. Tek. Ind., vol. 21, no. 1, pp. 1–10, 2022.
[33] J.-f. Chen, L. Wang, and Z.-p. Peng, "A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling," Swarm and Evolutionary Computation, vol. 50, p. 100557, 2019.
[34] F. Zhao, R. Ma, and L. Wang, "A Self-Learning Discrete Jaya Algorithm for Multiobjective Energy-Efficient Distributed No-Idle Flow-Shop Scheduling Problem in Heterogeneous Factory System," IEEE Transactions on Cybernetics, vol. 52, no. 12, pp. 12675-12686, 2022.

[35] D. M. Utama and M. D. Primayesti, "A novel hybrid Aquila optimizer for energy-efficient hybrid flow shop scheduling," Results Control Optim., vol. 9, p. 100177, 2022.
[36] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," Comput. Ind. Eng., vol. 158, p. 107408, 2021.
[37] C. Kumar and D. M. Mary, "Parameter estimation of three-diode solar photovoltaic model using an Improved-African Vultures optimization algorithm with Newton–Raphson method," J. Comput. Electron., vol. 20, no. 6, pp. 2563–2593, 2021.
[38] J. Zhang, M. Khayatnezhad, and N. Ghadimi, "Optimal model evaluation of the proton-exchange membrane fuel cells based on deep learning and modified African Vulture Optimization Algorithm," Energy Sources, Part A Recover. Util. Environ. Eff., vol. 44, no. 1, pp. 287–305, 2022.
[39] S. Mirjalili, "The ant lion optimizer," Adv. Eng. Softw., vol. 83, pp. 80–98, 2015.
[40] P. Yao and H. Wang, "Dynamic Adaptive Ant Lion Optimizer applied to route planning for unmanned aerial vehicle," Soft Comput., vol. 21, no. 18, pp. 5475–5488, 2017.
[41] J. Wang, P. Du, H. Lu, W. Yang, and T. Niu, "An improved grey model optimized by multi-objective ant lion optimization algorithm for annual electricity consumption forecasting," Appl. Soft Comput., vol. 72, pp. 321–337, 2018.
[42] D. M. Utama, T. Baroto, D. M. Maharani, F. R. Jannah, and R. A. Octaria, "Algoritma ant-lion optimizer untuk meminimasi emisi karbon pada penjadwalan flow shop dependent sequence set-up," J. Litbang Ind., vol. 9, no. 1, pp. 69–78, 2018.
[43] 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 June 2021, doi: 10.11591/eei.v10i3.2958, Jun. 2021. [Online]. Available: https://beei.org/index.php/EEI/article/view/2958
[44] D. M. Utama, "Minimizing Number of Tardy Jobs in Flow Shop Scheduling Using A Hybrid Whale Optimization Algorithm," Journal of Physics Conference Series, Mar. 2021, vol. 1845, p. 12017. doi: 10.1088/1742-6596/1845/1/012017.
[45] J. Carlier, "Ordonnancements a contraintes disjonctives," RAIRO-Operations Res., vol. 12, no. 4, pp. 333–350, 1978.
[46] C. R. Reeves, "A genetic algorithm for flowshop sequencing," Comput. Oper. Res., vol. 22, no. 1, pp. 5–13, 1995.
[47] Z. He, B. Tang, and F. Luan, "An Improved African Vulture Optimization Algorithm for Dual-Resource Constrained Multi-Objective Flexible Job Shop Scheduling Problems," Sensors, vol. 23, no. 1. 2023. doi: 10.3390/s23010090.
[48] H. Askr, M. A. Farag, A. E. Hassanien, V. Snášel, and T. A. Farrag, "Many-objective African vulture optimization algorithm: A novel approach for many-objective problems.," PLoS One, vol. 18, no. 5, p. e0284110, 2023, doi: 10.1371/journal.pone.0284110.
[49] J. Fan, Y. Li, and T. Wang, "An improved African vultures optimization algorithm based on tent chaotic mapping and time-varying mechanism.," PLoS One, vol. 16, no. 11, p. e0260725, 2021, doi: 10.1371/journal.pone.0260725.

AUTHORS BIOGRAPHY

Dana Marsetiya Utama is currently a Lecturer and a Researcher in Department of the Industrial Engineering, University of Muhammadiyah Malang indonesia. His research interests include optimization engineering, scheduling, production-inventory management, and modeling. He received the bachelor's degree in industrial engineering from Trunojoyo University, in 2008, and the master's degree in industrial engineering from the University of Brawijaya, in 2011. Recently, he received a Doctorate degree in Agroindustrial Technology from Brawijaya University in 2023. He can be contacted at email: dana@umm.ac.id.

Yolanda Mega Risma is currently a Student in the Department of Industrial Engineering, University of Muhammadiyah Malang indonesia. Her research interests include optimization engineering, scheduling, and modeling. She can be contacted at email: yolandamegarisma@gmail.com.

 DOI: 10.25077/josi.v22.n2.p126-141.2023