JURNAL ILMIAH TEKNIK INDUSTRI
ISSN: 1412-6869 (Print), ISSN: 2460-4038 (Online)
Journal homepage: http://journals.ums.ac.id/index.php/jiti/index
doi: 10.23917/jiti.v21i1.17634

An Energy-Efficient No Idle Permutations Flow Shop Scheduling Problem Using Grey Wolf Optimizer Algorithm

Cynthia Novel Al-Imron[1a◆], Dana Marsetiya Utama[1b], Shanty Kusuma Dewi[1c]

Abstract. *Energy consumption has become a significant issue in businesses. It is known that the industrial sector has consumed nearly half of the world's total energy consumption in some cases. This research aims to propose the Grey Wolf Optimizer (GWO) algorithm to minimize energy consumption in the No Idle Permutations Flowshop Problem (NIPFP). The GWO algorithm has four phases: initial population initialization, implementation of the Large Rank Value (LRV), grey wolf exploration, and exploitation. To determine the level of machine energy consumption, this study uses three different speed levels. To investigate this problem, 9 cases were used. The experiments show that it produces a massive amount of energy when a job is processed fast. Energy consumption is lower when machining at a slower speed. The performance of the GWO algorithm has been compared to that of the Cuckoo Search (CS) algorithm in several experiments. In tests, the Grey Wolf Optimizer (GWO) outperforms the Cuckoo Search (CS) algorithm.*

Keywords: *no idle permutation flow shop, energy efficiency, metaheuristic, grey wolf optimizer algorithm*

I. INTRODUCTION

The current energy crisis is one of the most pressing environmental issues that must be addressed (Utama et al., 2020). Nonrenewable energy resources are depleting, and greenhouse gases are rising due to overuse (Utama & Widodo, 2021). The industrial sector consumes roughly half of global energy consumption (Fang et al., 2011). The manufacturing sector consumes the vast majority of industrial energy. In 2011, China's industrial energy consumption accounted for 70.82 percent of total energy consumption, with the manufacturing sector accounting for 81.32 percent of total industrial energy consumption (Li & Lin, 2015). Manufacturing consumes 90% of industrial electrical energy in the United States. Based on this fact, manufacturers should implement measures to reduce energy consumption and carbon emissions.

Scheduling is one effort to improve energy efficiency in the manufacturing process (Mansouri et al., 2016). Energy-efficient scheduling (EES) has recently increased attention because it can reduce energy consumption without requiring additional equipment investment costs (Gahm et al., 2016).

Some manufacturing systems have no idle time as a constraint in production. These industries are glassmaking, integrated circuits, fibreglass processing, and ceramics. The No-Idle Permutation Flow Shop Problem (NIPFSP) is a permutation flow shop problem where the machine cannot be idle for an extended time (Shao et al., 2019). Due to no-idle constraints, the machine cannot be turned off during the manufacturing process in this NIPFSP problem (Zhao et al., 2021). As a result, no turn-off strategy can be used (Chen et al., 2019). This issue necessitates a significant amount of energy consumption. Researchers have presented several NIPFSP studies. Researchers typically use performance to reduce completion time and tardiness. Invasive weed optimization (Zhou et al., 2014), hybrid node and edge histogram (Shao et al., 2017), general variable neighbourhood search (Öztop et al., 2020), discrete bacterial memetic evolutionary algorithm (Agárdi et al., 2021), and integer linear programming (Croce, 2021) are among the algorithms proposed to reduce

[1] Industrial Engineering Department, Faculty of Engineering, Universitas Muhammadiyah Malang, Jalan Raya Tlogomas No. 246, Malang, 65144.

[a] email: cynthianovel88@gmail.com
[b] email: dana@umm.ac.id
[c] email: shanty@umm.ac.id
◆ corresponding author

Submited: 27-02-2022 Revised: 18-05-2022
Accepted: 08-06-2022

completion time. Several proposed procedures for reducing tardiness include bi-population estimation of distribution algorithm (Shen et al., 2015), cuckoo search (CS) (Sun & Gu, 2017), and hybrid discrete water wave (Zhao et al., 2020). Furthermore, Nagano, Rossi, and Martarelli (2019) proposed an integrated greedy method to minimize total flow time.

NIPFSP studies have been presented by researchers based on descriptions of previous studies. However, there has been no NIPFSP research to reduce energy consumption. In addition, several variations of the algorithm are proposed to solve this problem. As a result, sophisticated algorithms are required to solve efficient scheduling problems to minimize energy consumption. Research on EES NIPFSP is still scarce, which prompted us to investigate this issue by proposing the Grey Wolf Optimization (GWO) algorithm. The research proposes a GWO algorithm for solving the EES problem in the no-idle permutation flow shop problem (NIPFSP). The GWO algorithm is a metaheuristic algorithm that imitates the behaviour of the grey wolf in hunting its prey. The GWO algorithm was chosen because it has proven to be effective in solving various problems, including feature selection (Sathiyabhama et al., 2021), Building energy optimization (Ghalambaz et al., 2021), and. Travelling salesman problem (Panwar & Deep, 2021). This study aims to propose a GWO algorithm to determine the optimal parameter speed machine level to reduce energy consumption. In this study, we also attempt to determine the effect of machine speed level on energy consumption and computation time. The main contribution of this research is to propose a GWO algorithm to solve the no-idle permutation flow shop problem.

II. Research Method

Notations and Formulation of Mathematical models

This section describes the notation and formulation of the Mathematical model used in the NIPFSP problem. Figures 1 and 2 depict illustrations of the mutation flow shop scheduling (PFSP) and NIPFSP problems. In a typical PFSP problem, the machine can idle after completing a job (D. M. Utama, Baroto, & Widodo, 2020) (D. M. Utama, Garside, & Wicaksono, 2019). However, in Figure 2 of the NIPFSP problem, it is assumed that the machine is not allowed to idle after completing one job (Pan & Ruiz, 2014).

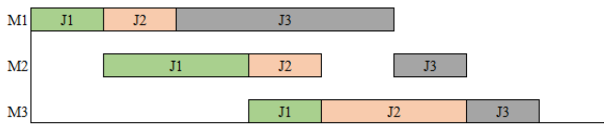

Figure 1. Illustration of a typical PFSP problem

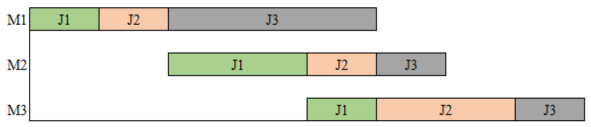

Figure 2. NIPFSP Problem Illustration

Several assumptions from the NIPFSP problem are used in this study. The assumptions used are (Li et al., 2021; Ruiz et al., 2009): (1) All sets of n jobs must be processed using the same process sequence on m machine sets. (2) At time 0, all jobs arrive and are ready to be processed. (3) To meet the no-idle requirement, the first job's processing start time on machines two through m must be delayed. (4) Each machine can only process one job at a time, and each job can only be processed once on each machine. (5) Once the first job begins to be processed, the process cannot be interrupted until the last job is completed. (6) Setup time is included in the job processing time. (7) No idle machines are permitted between jobs.

The notations used in the NIPFSP problem are as follows (Tasgetiren et al., 2013):

n	: Number of jobs
m	: Number of machines
i	: job index
j	: machine index
r	: speed level 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 time job i on machine j

C_{max}	: Makespan or completion time
μ_r	: Machine speed r during processing
τ_j	: Processing energy consumption on j machine
λ_r	: Machining speed r conversion factor
φ_j	: Energy consumption when machine j is idle
TEC	: Total energy consumption

A mathematical model formulation of the Mixed Integer Programming (MIP) model to minimize energy consumption in the NIPFSP problem is provided below.

Decision Variable

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

$$X_{ik} = \begin{cases} 1, \text{if job } i \text{ is the predecessor job of job } k \\ 0, \text{otherwise } (i<k) \end{cases}$$

Objective Function:

Min TEC (1)

Constrains:

$$\boldsymbol{C_{i,1} \geq \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r} \quad \forall_i = (1,..,n)} \quad (2)$$

$$\boldsymbol{C_{ij} - C_{i,j-1} \geq \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_j = (\mathbf{2},..,\mathbf{m}), \boldsymbol{i} = (\mathbf{2},..,\mathbf{n}) \quad (3)$$

$$\boldsymbol{C_{ij} - C_{kj} + DX_{ik} \geq \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}), \boldsymbol{k} = (\mathbf{1},..,\mathbf{n}) \quad (4)$$

$$\boldsymbol{C_{ij} - C_{kj} + DX_{ik} \leq D - \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}), \boldsymbol{k} = (\mathbf{1},..,\mathbf{n}) \quad (5)$$

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

$$\boldsymbol{\sum_{r=1}^{l} Y_{ijr} = 1} \quad \forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}) \quad (7)$$

$$\boldsymbol{Y_{ijr} = Y_{i,j+1,r}}$$

$$\forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}), \mathbf{r} = (\mathbf{1},..,\mathbf{l}) \quad (8)$$

$$\boldsymbol{\theta_j = C_{max} - \sum_{i=1}^{n} \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_j = (\mathbf{1},..,\mathbf{m}) \quad (9)$$

$$\textbf{TEC} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{r=1}^{l} \frac{P_{ij} \tau_j \lambda_r}{60 \mu_r} Y_{ijr} + \sum_{j=1}^{m} \frac{\varphi_j \theta_j \tau_j}{60} \quad (10)$$

$$\boldsymbol{S_j \leq C_{ij} - \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}) \quad (11)$$

$$\boldsymbol{F_j \geq C_{ij}} \quad \forall_i = (\mathbf{1},..,\mathbf{n}), \mathbf{j} = (\mathbf{1},..,\mathbf{m}) \quad (12)$$

$$\boldsymbol{F_j \geq S_j + \sum_{i=1}^{n} \sum_{r=1}^{l} \frac{P_{i1} Y_{ijr}}{\mu_r}}$$

$$\forall_i = (1,..,n), j = (1,..,m) \quad (13)$$

The objective function for TEC minimization is shown in equation (1). Constraint (2) displays the time required to complete each job on the first machine. Constraint (3) ensures that the next operation will be processed if the previous one has been completed. Constraints (4) and (5) depict the sequence of each job. Constraint (6) displays the calculation of the makespan (completion time). Constraints (7) and (8) ensure that each job is processed on all machines at the same machining speed. The idle time for each machine is shown in constraint (9). It is important to remember that idle time exists only at the start and end of the delay; there is no idle time between jobs. Constraint (10) is a total energy consumption calculation. Constraints (11), (12), and (13), respectively, ensure that no idle time occurs between jobs on each machine.

This problem is not difficult to solve in terms of NIPFSP scheduling. In this case, the researcher must understand when the machine can operate without being idle in between job processing. It is labelled as S_j, $j = (1,2,\ldots,m)$. Where , $S_1 = 0$. The machining start time formula is shown in equation (14).

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

$$j = (2,3,\ldots,m) \quad (14)$$

Equations (15) and (16) show how to calculate the completion time of each job, while equation (17) shows how to determine the makespan (completion time).

$$\boldsymbol{C_{(1),j} = S_j + p_{(1)}}, \quad \boldsymbol{j = (1,2,\ldots,m)} \quad (15)$$

$$\boldsymbol{C_{(i),j} = C_{(i-1),j} + p_{(j)}}$$

$$\boldsymbol{i = (2,3,\ldots,n), j = (1,2,\ldots,m)} \quad (16)$$

$$C_{max} = Max_{1\le h\le n}\left\{\sum_{i=1}^{h} C_{(i),j}\right\};\ j = (1,2,\dots,\mathrm{m}) \quad \textbf{(17)}$$

This study considers the machining speed factor when calculating the energy consumption generated when processing jobs. It is known that the machine operates at a speed of μ_rThe machine is not permitted to change its speed when processing jobs. When the machine is running at μ_r. The processing time for each operation is $O_{i,j} = \frac{p_{ij}}{\mu_r}$. The equation for calculating total energy consumption is presented (18) (Tasgetiren et al., 2013).

$$\mathrm{TEC}=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{r=1}^{l}\frac{P_{ij}\,\tau_j\lambda_r}{60\mu_r}\,Y_{ijr} + \sum_{j=1}^{m}\frac{\varphi_j\,\theta_j\,\tau_j}{60} \quad \textbf{(18)}$$

Proposed Algorithm

The researcher proposes the grey wolf optimizer to reduce energy consumption (GWO). The algorithm was inspired by a hunting grey wolf. The grey wolf is a social animal that prefers to live in groups of 5-12 tails. The proposed algorithm is based on the hunting behaviour of the grey wolf (Mirjalili et al., 2014). The group is divided into four roles: alpha (α), beta (β), delta (δ) and omega (ω). Algorithm 1 (Mirjalili et al., 2014) contains the GWO algorithm's pseudocode. The grey wolf optimizer algorithm is divided into four stages: initial population initialization, Large Rank Value (LRV) implementation, grey wolf exploration, and exploitation.

The first stage is the initialization population. The initial position and GWO parameters are set at this stage. The parameters used are the number of grey wolves and the number of iterations. The second stage involves converting GWO positions to permutation sequences using the Large Rank Value (LRV) principle. The LRV method is a simple procedure that effectively converts a herd's position to a permutation job sequence (Utama, 2019) (Widodo & Utama, 2021). Figure 3 depicts the LRV method. LRV is an efficient method for mapping continuous values into permutation jobs (Utama, 2021).

The third stage is the grey wolf exploration stage. Based on the hunting behaviour of the grey wolf, there are three stages of hunting. These stages are tracking, siege, and attack, all of which are carried out for optimization. These behaviours are expressed using equations (19), (20), (21), and (22) as follows:

$$\vec{D} = \left|\vec{C}\cdot\vec{X_p}\,(t) - \vec{X}\,(t)\right| \quad \textbf{(19)}$$

$$\vec{X}\,(t+1) = \vec{X_p}\,(t) - \vec{A}\cdot\vec{D} \quad \textbf{(20)}$$

$$\vec{A} = 2\vec{\alpha}\cdot\vec{r_1} - \vec{\alpha} \quad \textbf{(21)}$$

$$\vec{C} = 2\cdot\vec{r_2} \quad \textbf{(22)}$$

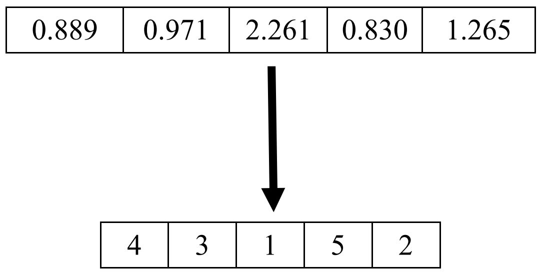

Figure 3. Application of LRV

Where t denotes the loop (iteration). $\overrightarrow{X_p}$ is the prey's position vector, and $\vec{X}$ is the grey wolf's position vector. Vector coefficients are represented by the $\vec{D}$, $\vec{A}$, dan $\vec{C}$. During iterations of mathematical modeling of wolves catching prey, the value a decreases linearly from 2 to 0. At the same time, r1 and r2 are random vectors with values ranging from 0 to 1.

The fourth stage is the exploitation stage, in which hunting is led by and occasionally hunted. According to the social hierarchy, α is the best solution candidate, followed by β and δ. Hunting is represented in finding the optimal position by equations (23-27) as follows:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}\cdot\overrightarrow{X_\alpha} - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}\cdot\overrightarrow{X_\beta} - \vec{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}\cdot\overrightarrow{X_\delta} - \vec{X}\right| \quad \textbf{(23)}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}\cdot\left(\overrightarrow{D_\alpha}\right) \quad \textbf{(24)}$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}\cdot\left(\overrightarrow{D_\beta}\right) \quad \textbf{(25)}$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}\cdot\left(\overrightarrow{D_\delta}\right) \quad \textbf{(26)}$$

$$\vec{X}\,(\mathrm{t}+1) = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \quad \textbf{(27)}$$

Data and experimental procedure

The processing time derived from Tailard (Taillard, 1993) research is used to present nine job and machine combinations in this study. Table

1 shows the data and the combination of jobs and machines used for research.

Algorithm 1: Pseudocode of Grey Wolf Optimizer (GWO) Algorithm

```
Initialize the grey wolf position
Initialize $\alpha, A$, and $C$
Use LRV to convert job permutation sequences
Compute the fitness of each search agent
Set the $X_\alpha$, $X_\beta$, $X_\delta$ according to the fitness
t=1 While(t<Max)
          for each wolf
                Update the position by equation (27)
          End for
          Update $\alpha, A$, and $C$
          Use LRV to convert job permutation sequences
          Compute the fitness of each search agent
          Update the $X_\alpha$, $X_\beta$, $X_\delta$
          fitnesst=t+1
end while
Output $X_\alpha$
```

Table 1. Research data

Problem	Job and Machine	Speed Level	References
Case 1	20 job 5 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 2	20 job 10 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 3	20 job 20 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 4	50 job 5 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 5	50 job 10 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 6	50 job 20 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 7	100 job 5 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 8	100 job 10 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)
Case 9	100 job 20 Machine	Fast, Normal, Low	Tailard (Taillard, 1993)

This researcher employed the GWO algorithm's iteration parameters of 500 iterations and 500 populations. The researcher employs these parameters because the larger the population and the number of iterations, the more and better solutions will be produced (D. Utama, 2021). Researchers also employed the machine speed factor, which was classified into three categories: fast, normal, and low. The researcher assumed that each job would be processed at the same machining speed on all machines in this experiment. No speed changes will be permitted during job processing. The machining speed data set, which includes low, medium, and high speeds, is μ_r = (0.8, 1, 1.2).

Researchers used the change factor λ_r = (0.6, 1, 1.5) to estimate low, medium, and high energy consumption during the process. The power of all machines is the same, namely τ_j = 60 kW and φ_j = 0.05 kW (Mansouri et al., 2016). This experiment was repeated 27 times. Furthermore, the performance of the proposed algorithm, namely the Gray Wolf Optimizer (GWO), was evaluated by comparing it to the Cuckoo Search Algorithm (CS). The CS algorithm is also run through 500 iterations on a population. For each trial, the computation time is also recorded. The study was carried out on a Windows 10 Intel (R) Core (TM) i3-2348M CPU RAM 2 Gb with Matlab R2018a software. The researcher employed the Efficiency Index (EI) to assess the performance of the GWO algorithm, which is presented in equation form (28). EI is a simple method for assessing algorithm performance. Furthermore, the Wilcoxon test is presented to evaluate the algorithm's performance.

$$\text{EI} = \frac{proposed\ algorithm}{another\ algorithm} \times 100\ \% \qquad \textbf{(28)}$$

III. RESULT AND DISCUSSION

Influence speed level to TEC and computation time

The experimental results of the effect of speed level on TEC are described in this section. This study applied a high, normal, and low-speed machine level for each machine. As a result, the processing time of jobs changes based on the speed level. We assume that the speed level is the same on all machines.

Table 2. Experimental results Influence speed level to TEC on the GWO algorithm (kW)

Problem	Speed level		
	Fast	Normal	Slow
Case 1	6478,375	5198,05	3920,8125
Case 2	13152,7917	10617	8112,5
Case 3	26349,125	21504	16620,75
Case 4	15185,7917	12184	9192,0625
Case 5	31602,4583	25360	19143,9375
Case 6	66057,7083	53306,55	40701,6875
Case 7	32336,5	25906,15	19484,8125
Case 8	65240,2917	52322,5	39458,5625
Case 9	125272,417	100936,3	76758,5625

Table 3. Experimental results Influence speed level to TEC on the CS algorithm (kW)

Problem	Speed level		
	Fast	Normal	Slow
Case 1	6499,458	5225,6	3956,625
Case 2	13196,29	10648,25	8171,938
Case 3	26538,21	21721,35	16924,81
Case 4	15212,33	12206,4	9228,625
Case 5	31684,83	25498,35	19309,88
Case 6	66072,5	53738	41168,81
Case 7	32360,46	25.956	19533,63
Case 8	65381,5	52466,85	39637,81
Case 9	125307,1	101162,6	76915,13

The Influence speed level to the TEC experiment on the GWO algorithm yielded the results in Table 2. The experimental results show that the resulting energy consumption is high if the job is processed fast. On the other hand, the energy consumption will be lower if the job is processed at a slow machining speed. Table 3 displays the experimental results of the Influence speed level to TEC on the CS algorithm. The CS algorithm research results are comparable to the GWO algorithm. When a task is completed fast, a significant amount of energy is consumed. Slower machining speeds, on the other hand, use less energy. These findings suggest that the machining speed affects the amount of energy consumed.

Table 4. Computing Time Results on the GWO Algorithm (second)

Problem	Speed level		
	Fast	Normal	Slow
Case 1	567,342275	538,35833	685,564997
Case 2	655,129263	648,821206	608,869773
Case 3	618,569987	648,621384	6952,285527
Case 4	716,624276	707,136529	782,505394
Case 5	760,899965	702,117919	731,870592
Case 6	821,376432	843,641038	875,560132
Case 7	955,431533	954,772712	946,290208
Case 8	989,634884	933,012641	944,369527
Case 9	918,586824	986,968063	996,40119

Tabel 5. Computing Time Results on the Cuckoo Search Algorithm (second)

Problem	Speed level		
	Fast	Normal	Slow
Case 1	1096,435	1363,35	1993,777
Case 2	1195,597	1060,853	1886,419
Case 3	1454,854	1592,74	2690,86
Case 4	1024,306	1107,559	1969,622
Case 5	2054,782	2585,024	2976,324
Case 6	2922,833	3295,6	4096,848
Case 7	2264,805	2043,387	3078,926
Case 8	2712,985	3545,277	3896,256
Case 9	3001,97	3575,074	3496,295

Tables 4 and 5 show the computational time required to solve the NIPFSP problem case using the proposed GWO and CS algorithms. The results show that the computational time

Table 6. Comparison of Algorithms

Problem	Job and Machine	Efficiency Index (EI) %					
		GWO			CS		
		Fast	Normal	Slow	Fast	Normal	Slow
Case 1	20 job 5 Machine	100	100	100	99,676%	99,627%	99,095%
Case 2	20 job 10 Machine	100	100	100	99,688%	99,707%	99,273%
Case 3	20 job 20 Machine	100	100	100	99,288%	98,999%	98,785%
Case 4	50 job 5 Machine	100	100	100	99,826%	99,816%	99,604%
Case 5	50 job 10 Machine	100	100	100	99,740%	99,457%	99,141%
Case 6	50 job 20 Machine	100	100	100	99,978%	99,197%	98,865%
Case 7	100 job 5 Machine	100	100	100	99,926%	99,808%	99,750%
Case 8	100 job 10 Machine	100	100	100	99,784%	99,725%	99,548%
Case 9	100 job 20 Machine	100	100	100	99,707%	99,462%	99,176%
	Average	100	100	100	100	99,533%	99,249%

generated by each case has a varying computation time. The comparison of computational time at three-speed levels of the GWO algorithm in Table 4 shows that the computational time at each speed level is relatively the same. Similarly, when comparing the computational time at three-speed levels of the CS algorithm (Table 5). Another finding from the computational time results is that the more job and machine combinations used, the greater the computational time generated. Furthermore, the GWO algorithm has a faster computation time than the CS algorithm.

Comparison Algorithm

We describe the comparison algorithm in this section. The performance of the proposed GWO algorithm compared to the comparison algorithm is measured using the EI presented in equation (26), as explained in the experimental procedure section. Table 6 displays the results of comparing the efficiency index values of the GWO and CS algorithms. The proposed GWO algorithm outperforms the cuckoo search algorithm according to the results. This result demonstrated that the GWO algorithm outperforms CS at all three-speed levels, with an EI value of 100%. Furthermore, the EI value of the CS algorithm is 99.735% for fast speeds, 99.533% for normal speeds, and 99.249% for slow speeds.

A comparison of the algorithms was also performed using the Wilcoxon test to compare the performance of the GWO algorithm with the cuckoo search algorithm. The Wilcoxon test results are shown in Table 7. According to the Wilcoxon test results, the asymp value Sig. (2-tailed) is less than 0.05. It is possible to conclude that the grey wolf optimizer and cuckoo search algorithms perform differently. The results revealed that the GWO algorithm outperforms the cuckoo search algorithm. It indicates that the proposed GWO algorithm outperforms cuckoo search statistically.

Managerial Implications

According to the findings of this study, the speed of the machine level has a significant factor in energy use when processing jobs. Efficient use of energy also helps companies save on the cost of machinery produced and helps reduce the impact of global warming. It also reduces the harmful effects of the scarcity of energy resources (Huang, Pan, Huang, Suganthan, & Gao, 2021). Therefore, the production planner needs to schedule the right job to reduce energy consumption. This research applies the speed-scaling method to solve the problem of energy consumption in an efficient way to help decision making. Using energy-efficient scheduling, this

Table 7. Wilcoxon test

Test	Z	Asymp. Sig. (2-tailed)
GWO-CS Fast	-2.555	0.011
GWO-CS Normal	-2.673	0.008
GWO-CS Slow	-2.666	0.008

research is proven to reduce energy consumption so that it can impact the company's economic efficiency.

This study found a relationship between speed level and energy consumption. This study indicates that fast machining speed requires a large amount of energy to process a job. Meanwhile, low machining speed requires little energy. The machining speed affects the completion time of a job (J.-f. W. Chen, Lin Peng, Zhi-ping 2019).

When using a fast machining speed level, job completion time is faster. However, the energy required for job processing was extremely high. Conversely, job completion time is longer when using a slow machining speed. However, the energy required for job processing is low (Fang, Luo, & Che, 2021; M Fatih Tasgetiren, Öztop, & Pan, 2019). Therefore, the production planner can decide in a no-idle permutation flow shop regarding production efficiency and energy consumption considerations using the proposed approach.

This study also found a relationship between energy consumption and the number of machines used in processing. The results show that the more machines are used to process a job, the greater the energy produced. Some companies with identical items can divide the job into sub lots to minimise the job completion time (Fang et al., 2021). The lot streaming method used to divide a job into sub lots can be considered for production planners if they want to minimize energy consumption, but the company has many machines operating.

IV. CONCLUSION

In this study, researchers attempted to solve the No-Idle Permutation Flow Shop Problem (NIPFSP) while minimizing energy consumption. The grey wolf optimizer algorithm is proposed in this study to solve this problem. The experiments show that if the job is processed fast, the energy consumption is high. Conversely, if the job is processed at a slow-speed level, lower energy consumption. The results show that the grey wolf optimizer algorithm admirably solves the NIPFSP scheduling problem. The proposed algorithm was also compared to a comparison algorithm to determine the performance of this grey wolf optimizer algorithm. According to the results of the experiments, the grey wolf optimizer algorithm outperforms the cuckoo search algorithm.

The proposed metaheuristic could be used to solve other manufacturing scheduling problems with different limitations in the future, such as no-wait or blocked flow shops. Several objectives such as total flowtime, tardiness, and tardy jobs will be explored for future research. This study has limitations, such as ignoring the NIPFSP problem's setup and removal times. More research into the setup and removal times of NIPFSP problems is necessary.

REFERENCES

Agárdi, A., Nehéz, K., Hornyák, O., & Kóczy, L. T. (2021). A Hybrid Discrete Bacterial Memetic Algorithm with Simulated Annealing for Optimization of the Flow Shop Scheduling Problem. Symmetry, 13(7), 1131.

Chen, J.-f., Wang, L., & Peng, Z.-p. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm and Evolutionary Computation, 50, 100557. doi:https://doi.org/10.1016/j.swevo.2019.100557

Chen, J.-f. W., Lin Peng, Zhi-ping (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evolutionary Computation, 50, 100557.

Della Croce, F. G., Andrea Salassa, Fabio (2021). Minimizing total completion time in the two-machine no-idle no-wait flow shop problem. Journal of Heuristics, 27(1), 159-173.

Fang, K., Luo, W., & Che, A. (2021). Speed scaling in two-machine lot-streaming flow shops with consistent sublots. Journal of the Operational Research Society, 72(11), 2429-2441.

Fang, K., Uhan, N., Zhao, F., & Sutherland, J. W. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. Journal of Manufacturing Systems, 30(4), 234-240. doi:https://doi.org/10.1016/j.jmsy.2011.08.004

Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. European Journal of Operational Research, 248(3), 744-757. doi:https://doi.org/10.1016/j.ejor.2015.07.017

Ghalambaz, M., Jalilzadeh Yengejeh, R., & Davami, A. H. (2021). Building energy optimization using Grey Wolf Optimizer (GWO). Case Studies in Thermal Engineering, 27, 101250. doi:https://doi.org/10.1016/j.csite.2021.101250

Huang, Y.-Y., Pan, Q.-K., Huang, J.-P., Suganthan, P. N., & Gao, L. (2021). An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. Computers Industrial Engineering, 152, 107021.

Li, K., & Lin, B. (2015). The efficiency improvement potential for coal, oil and electricity in China's manufacturing sectors. Energy, 86, 403-413. doi:https://doi.org/10.1016/j.energy.2015.04.013

Li, Y.-Z., Pan, Q.-K., Li, J.-Q., Gao, L., & Tasgetiren, M. F. (2021). An Adaptive Iterated Greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems. Swarm Evolutionary Computation, 63, 100874.

Mansouri, S. A., Aktas, E., & Besikci, U. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. European Journal of Operational Research, 248(3), 772-788. doi:https://doi.org/10.1016/j.ejor.2015.08.064

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

Nagano, M. S., Rossi, F. L., & Martarelli, N. J. (2019). High-performing heuristics to minimize flowtime in no-idle permutation flowshop. Engineering Optimization, 51(2), 185-198.

Öztop, H., Tasgetiren, M. F., Kandiller, L., & Pan, Q.-K. (2020). A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling. Paper presented at the 2020 IEEE Congress on Evolutionary Computation (CEC).

Pan, Q.-K., & Ruiz, R. (2014). An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem. Omega, 44, 41-50.

Panwar, K., & Deep, K. (2021). Discrete Grey Wolf Optimizer for symmetric travelling salesman problem. Applied Soft Computing, 105, 107298. doi:https://doi.org/10.1016/j.asoc.2021.107298

Ruiz, R., Vallada, E., & Fernández-Martínez, C. (2009). Scheduling in flowshops with no-idle machines. In Computational intelligence in flow shop and job shop scheduling (pp. 21-51): Springer.

Sathiyabhama, B., Kumar, S. U., Jayanthi, J., Sathiya, T., Ilavarasi, A. K., Yuvarajan, V., & Gopikrishna, K. (2021). A novel feature selection framework based on grey wolf optimizer for mammogram image analysis. Neural Computing and Applications, 33(21), 14583-14602.

Shao, W., Pi, D., & Shao, Z. (2017). Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. Applied Soft Computing, 54, 164-182.

Shao, W., Pi, D., & Shao, Z. (2019). Local Search Methods for a Distributed Assembly No-Idle Flow Shop Scheduling Problem. IEEE Systems Journal, 13(2), 1945-1956. doi:10.1109/JSYST.2018.2825337

Shen, J.-n., Wang, L., & Wang, S.-y. (2015). A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. Knowledge-Based Systems, 74, 167-175.

Sun, Z., & Gu, X. (2017). 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, 9(6), 953.

Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research, 64(2), 278-285.

Tasgetiren, M. F., Öztop, H. G., Liang , & Pan, Q.-K. L., Xinyu. (2019). A variable iterated local search algorithm for energy-efficient no-idle flowshop scheduling problem. Procedia Manufacturing, 39, 1185-1193.

Tasgetiren, M. F., Pan, Q.-K., Suganthan, P. N., & Buyukdagli, O. (2013). A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. Computers Operations Research, 40(7), 1729-1743.

Utama, D. (2021). Minimizing Number of Tardy Jobs in Flow Shop Scheduling Using A Hybrid Whale Optimization Algorithm. Paper presented at the Journal of Physics: Conference Series.

Utama, D. M. (2019). Salp Swarm Algorithm Untuk Meminimasi Konsumsi Energi Pada Penjadwalan Flow Shop Dengan Set Up Dan Removal Time. Prosiding SENTRA (Seminar Teknologi dan Rekayasa)(5), 79-85.

Utama, D. M., Baroto, T., & Widodo, D. S. (2020). Energy-efficient flow shop scheduling using hybrid grasshopper algorithm optimization. Jurnal Ilmiah Teknik Industri, 19(1), 30-38.

Utama, D. M., Garside, A. K., & Wicaksono, W. (2019). Pengembangan Algoritma Hybrid Flowshop Three-Stage Dengan Mempertimbangkan Waktu Setup. Jurnal Ilmiah Teknik Industri, 18(1), 72-78.

Utama, D. M., & Widodo, D. S. (2021). An energy-efficient flow shop scheduling using hybrid Harris hawks optimization. Bulletin of Electrical Engineering and Informatics, 10(3), 1154-1163.

Utama, D. M., Widodo, D. S., Ibrahim, M. F., Hidayat, K., Baroto, T., & Yurifah, A. (2020). The hybrid whale optimization algorithm: A new metaheuristic algorithm for energy-efficient on flow shop with dependent sequence setup. Journal of Physics: Conference Series, 1569(2), 022094. doi:10.1088/1742-6596/1569/2/022094

Widodo, D. S., & Utama, D. M. (2020). The hybrid ant lion optimization flow shop scheduling problem for minimizing completion time.

Zhao, F., Ma, R., & Wang, L. (2021). 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, 1-12. doi:10.1109/TCYB.2021.3086181

Zhao, F., Zhang, L., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2020). A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. Expert Systems with Applications, 146, 113166.

Zhou, Y., Chen, H., & Zhou, G. (2014). Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. Neurocomputing, 137, 285-292.