Engineering and Applied Science Research
https://www.tci-thaijo.org/index.php/easr/index
Published by the Faculty of Engineering, Khon Kaen University, Thailand

A novel Tiki-Taka algorithm to optimize hybrid flow shop scheduling with energy consumption

Mohd Fadzil Faisae Ab Rashid*[1] and Muhammad Ammar Nik Mu'tasim[2]

[1]Department of Industrial Engineering, College of Engineering, Universiti Malaysia Pahang, 26300, Kuantan, Pahang, Malaysia
[2]Faculty of Mechanical and Automotive Engineering Technology, Universiti Malaysia Pahang, 26600, Pekan, Pahang, Malaysia

Received 18 January 2021
Revised 24 May 2021
Accepted 2 June 2021

Abstract

Hybrid flow shop scheduling (HFS) has been thoroughly studied due to its significant impact on productivity. Besides the impact on productivity, the abovementioned problem has attracted researchers from different background because of its difficulty in obtaining the most optimum solution. HFS complexity provides good opportunity for researcher to propose an efficient optimization method for the said problem. Recently, research in HFS has moved towards sustainability by considering energy utilization in the study. Consequently, the problem becomes more difficult to be solved via existing approach. This paper modeled and optimized HFS with energy consumption using Tiki-Taka Algorithm (TTA). TTA is a novel algorithm inspired by football playing style that focuses on short passing and player positioning. In different with existing metaheuristics, the TTA collected information from nearby solution and utilized multiple leaders' concept in the algorithm. The research began with problem modeling, followed by TTA algorithm formulation. A computational experiment is then conducted using benchmark problems. Then, a case study problem is presented to assess the applicability of model and algorithm in real-life problems. The results indicated that the TTA consistently was in the first and second ranks in all benchmark problems. In addition, the case study results confirmed that TTA is able to search the best fitness solution by compromising the makespan and total energy utilization in the production schedule. In future, the potential of TTA will be further investigated for flexible hybrid flow shop scheduling problems.

Keywords: Hybrid flow shop, Tiki-Taka algorithm, Energy efficiency

Nomenclature

n	Total number of jobs	t_{iks}	Processing time of job i on machine k at stage s
i	Index for job. $i = 1, 2, \dots, n$	E_{ks}	Power rate of machine k at stage s in Watt
S	Total number of stage	C_i	Completion time of job i
s	Index for stage. $s = 1, 2, \dots, S$	C_{max}	Makespan time
K_s	Total number of machines at stage S	TEC	Total energy consumption
k_s	Index for machine at stage $S. k_s = 1, 2, \dots, K_s$.		

1. Introduction

Production scheduling is an important activity in manufacturing. This activity refers to assigning resources such as plant, machine, process, material, and human, among others, to produce finish or semi-finish products. The production scheduling problem is mainly divided into flow shop and job shop scheduling problems, depending on the resource flows. One of the production scheduling variants is known as hybrid flow shop scheduling (HFS).

HFS is the scheduling problem in which the production is conducted in a few stages. In each stage, parallel machines exist [1]. The basic version of HFS assumes that the process must strictly follow the predetermined process precedence, while the machine has similar capacities in performing the task. Consequently, there are two main issues in HFS: to determine the job processing sequence, and to assign job to specific machine in each stage [2].

Various types of HFS have been studied in previous researches. The most basic version is known as identical parallel machine, where all machines in every stage are identical. The second version is uniform parallel machine, in which the processing time is dependent on the machine speed. Meanwhile, the third version is the unrelated parallel machine; in this version, the parallel machine does not depending on another machine. The processing time for similar process might be different because of model variation [3]. This research focuses on unrelated HFS problem.

Recently, research in HFS tend to consider energy utilization, besides the common aims such as total completion time, lateness and penalty. The energy consideration not only provides sustainable advantage, but also reduces energy cost for manufacturer. Li et al. modeled the energy aware HFS by considering the energy utilization in standby and processing states. In addition, their work considered the setup time between consecutive jobs [4].

*Corresponding author.
Email address: ffaisae@ump.edu.my
doi: 10.14456/easr.2022.21

On the other hand, researcher embeds the energy utilization in HFS by minimizing the total production cost [5]. This approach, however, is only effective when the energy cost contributes to a certain portion from the total cost. When the energy cost is relatively small compared with other costs, the energy saving effect will be insignificant and manufacturer tends to ignore this solution. In order to avoid this situation, researcher would optimize HFS with energy utilization by lowering the relative importance of energy in the optimization process [6]. This leads the obtained solution to have preference over the makespan and tardiness objectives.
Another strategy used in HFS to reduce energy utilization is by manipulating the energy price. In many countries, the electricity rate is different during peak and off-peak period. In this approach, the total energy consumption is still the same, but the production schedule is shifted to the off-peak period with lower electricity rate [7, 8]. As such, the energy cost borne by the company is reduced, although the carbon footprint will remain the same.

Various types of metaheuristic algorithms were proposed to optimize HFS with energy utilization for the past three years (Table 1). Among popular optimization algorithm for HFS with energy consumption is Evolutionary Algorithm (EA). Jiang and Zhang improved multi-objective EA by introducing external archive population for convergence and local search for population diversity [9]. Meanwhile, Liu et. al used weighted sum approach in EA to handle makespan and energy objectives in optimization [10]. On the other hand, Gong et. al hybridized EA with variable neighborhood search (VNS) to enhance the algorithm performance [11].

Table 1 Metaheuristic algorithms used in HFS with energy utilization from 2019

Metaheuristic algorithm	**Reference**
Evolutionary algorithm	Jiang & Zhang, 2019 [9].
	Liu et al., 2020 [10].
	Gong et al., 2020 [11].
Genetic algorithm	Schulz, 2019 [5]
	Meng et al., 2019 [12]
Imperialist competitive algorithm	Li et al., 2019 [6]
	Zhou et al., 2019 [13]
	Tao et al, 2020 [14]
Ant lion optimization	Geng et al., 2020 [8]
Whale optimization algorithm	Utama et al., 2020 [15]
Multi-verse optimizer	Geng et al, 2019 [16]
Grasshopper algorithm optimization	Utama et al., 2020 [17]

Another algorithm implemented for HFS with energy consumption is Genetic Algorithm (GA). This algorithm has been improved by introducing energy-conscious decoding method that reduces energy consumption in the solution decoding [12]. Another work utilizing GA is by introducing intelligent swap and shifting procedure in Genetic Algorithm (GA) [5]. It was reported that this approach is able to find near-optimal solutions.

In 2019 and 2020, it was claimed that Imperialist Competitive Algorithm (ICA) is one of the frequently used algorithms to optimize HFS with energy consumption. Zhou et al., for example, modified the ICA by clustering the candidate solution into groups based on the solution fitness [13]. The abovementioned researchers found that this approach was able to keep solution diversity. On the other hand, Li et al. presented two-level ICA [6]; the first level consists of the strongest candidate solutions, while the second level considers other solutions. Meanwhile, Tao et al. proposed a discrete version of ICA [14]. This approach requires special operators to conduct mathematical operation in the algorithm.

Besides popular algorithms, researchers also implemented relatively new algorithms to optimize HFS with energy consumption. For instance, Geng et al. utilized Ant Lion Optimization (ALO) to optimize the problem. ALO was proposed back in 2014 to mimic the hunting behavior of ant lions. The results indicated that the multi-objective ALO was comparable with well-established algorithms such as NSGA-II and multi-objective PSO [8]. Apart from ALO, Whale Optimization Algorithm (WOA) is another new algorithm used to optimized HFS with energy utilization [15]. For optimization purposes, the WOA was hybridized with local search strategies to enhance the algorithm performance, in which the hybrid WOA was found to have successfully produced optimum energy consumption while maintaining the completion time.

Multi-Verse Optimizer (MVO) is also a relatively new algorithm implemented to optimize HFS with energy utilization [16]. The MVO is improved by embedding additional operators such as Lèvy Flight and local search to improve the performance. Grasshopper Algorithm Optimization (GOA) is also utilized to optimize the HFS with energy problem in printing company [17]. The optimization algorithm is able to reduce energy utilization in a company.

Based on the previous research, variety of optimization algorithms were implemented to optimize the HFS with energy utilization. One of the reasons for algorithm variation is that none of the single algorithm is able to perform best in all conditions. In other words, the performance of optimization algorithm depends on numerous factors, such as the algorithm coding, problem representation, computational experiment environment, and test problem/case study used during the experiment. Apart from adopting well-established algorithms such as EA and GA, researchers also tend to utilize relatively new algorithms, such as WOA, MVO and GOA. These algorithm performances are explored for the studied problem.

This research, therefore, proposes a novel algorithm called Tiki-Taka Algorithm (TTA) to optimize HFS with energy utilization. TTA is an optimization algorithm inspired by football playing style called tiki-taka. In this playing style, the player makes a short pass to the nearby player while aiming to deliver the ball to the key players. Using the same concept, the TTA gathers information from a nearby solution and the leading solutions in order to progress to the next iteration. The next section presents HFS with energy utilization problem, followed with TTA details. Next, this paper discusses results from computational experiment, as well as the case study. Finally, the conclusion is made based on the findings.

2. Hybrid flow shop with energy consumption problem

Hybrid flow shop scheduling (HFS) combines parallel machine and flow shop scheduling problem. In HFS, there are n jobs to be processed at S stages in a predetermined order. In each stage, s, there are K_s parallel and unrelated machines that can be used to conduct similar process. However, the capabilities of each machine in a stage differ due to model and speed variants. There are two main issues to be solved in HFS problem; the first issue is the sequence of job processing, whereas the second issue is the job-machine assignment in a particular stage. The following notations are used for HFS modeling.

The optimization objective in equation (1) aims to minimize the completion time, which is also known as makespan. This formula refers to the completion time of the last operation in the whole planning period. The second objective function (2) calculates the total energy utilization. TEC only calculates the energy utilization during operational period because in normal practice, the idle machine will be turned off for safety reasons.

$$C_{\max} = \max\left\{C_i\right\} \quad i = 1, 2, ..., n \tag{1}$$

$$TEC = \sum_{i=1}^{n} \sum_{s=1}^{S} \sum_{ks=1}^{K_S} t_{iks} \cdot E_{ks} \cdot y_{ik_S} \tag{2}$$

Subjected to:

$$y_{ik_S} \begin{cases} 1 & \text{if job } i \text{ is processed at machine } k_S \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\sum_{k_S=1}^{K_S} y_{ik_S} = 1, \quad \forall i = 1, 2, ..., n; \quad s = 1, 2.., S \tag{4}$$

Constraint (4) ensures that the job, i, is only processed at exactly one machine in each stage. The HFS problem is represented using continuous representation in a matrix of $n \times \sum_{s=1}^{S} K_s$, as in equation (5). Equation (6) presents the minimum value in each row. The job sequence will be determined by sorting X_{imin} in an ascending order, as per equation (7).

$$X = \begin{bmatrix} x_{1,1,1} & x_{1,1,2} & \cdots & x_{1,S,K_S} \\ x_{2,1,1} & x_{2,1,2} & \cdots & x_{2,S,K_S} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n,1,1} & x_{n,1,2} & \cdots & x_{n,S,K_S} \end{bmatrix} \tag{5}$$

$$X_{i_{\min}} = \begin{bmatrix} x_{1_{\min}} \\ x_{2_{\min}} \\ \cdots \\ x_{n_{\min}} \end{bmatrix}$$
(6)

$$X'_j = \text{ascending}\left\{X_{i_{\min}}\right\} \tag{7}$$

$$J = \{j\}; \quad \forall j = 1, 2, ..., n \tag{8}$$

2.1 Numerical example of problem modeling

As an example, the HFS model is let to consider a three-job and two-stage problem. Table 2 represents the processing time for the jobs at different machines.

Table 2 Example of HFS processing time

Job	Stage 1		Stage 2		
	M1_1	**M1_2**	**M2_1**	**M2_2**	**M2_3**
J1	8	11	3	5	3
J2	6	4	7	9	6
J3	2	3	6	4	4

2.1.1 Solution representation

The HFS problem is represented using continuous encoding in a matrix of $n \times \sum_{s=1}^{S} K_s$. In initial iteration, the problem representation is randomly created, while for the rest of iteration, the problem representation is generated from new player position in TTA. For the abovementioned example problem, the problem is represented in Table 3.

Table 3 Example of problem representation

Job	m_{1_1}	m_{1_2}	m_{2_1}	m_{2_2}	m_{2_3}
j_1	0.7174	0.1188	0.2760	0.3669	0.9024
j_2	0.2228	0.0890	0.9993	0.4492	0.0692
j_3	0.1620	0.6740	0.6966	0.6724	0.5395

To decode the job processing sequence, a minimum value from each row that represents the jobs is extracted. Then, the minimum value in each row is sorted in an ascending order. Using this rule, the decoded job processing sequence will be j_2, j_1, and j_3 (Table 4).

Table 4 (a) Minimum representation value, (b) Sorted job according to minimum representation

Job	Minimum	Job	Minimum
j_1	0.1188	j_2	0.0692
j_2	0.0692	j_1	0.1188
j_3	0.1620	j_3	0.1620

The next decision is to assign jobs to a specific machine in the stages. For this purpose, the similar rule that is based on minimum representation value is adopted. For job j_1 in the first stage, the representation value of m_{1_2} is smaller than m_{1_1}. Therefore, the j_1 in Stage 1 will be assigned to m_{1_2}. On the other hand, j_1 in Stage 2 will be assigned to m_{2_1} because the representation value for m_{2_1} is the smallest, compared with m_{2_2} or m_{2_3}. The job-machine assignment for this example is presented in Table 5.

Table 5 Job-machine assignment

Job	Assigned machine	
	Stage 1	Stage 2
j_2	m_{1_2}	m_{2_3}
j_1	m_{1_2}	m_{2_1}
j_3	m_{1_1}	m_{2_3}

Figure 1 show a schedule generated from the decoded HFS solution in the presented example. The job in a specific stage will be assigned to the machine subjected to the machine availability and also precedence between stages. For example, j_3 is the only job that being assigned to m_{1_1} in Stage 1. Although j_3 is the last job to be processed, since the machine m_{1_1} is available after j_2 and j_1 in Stage 1 were assigned, it will begin the process from time 0. In contrast, j_3 on the second stage can only begin the process at the 10th minute due to availability of m_{2_3}.

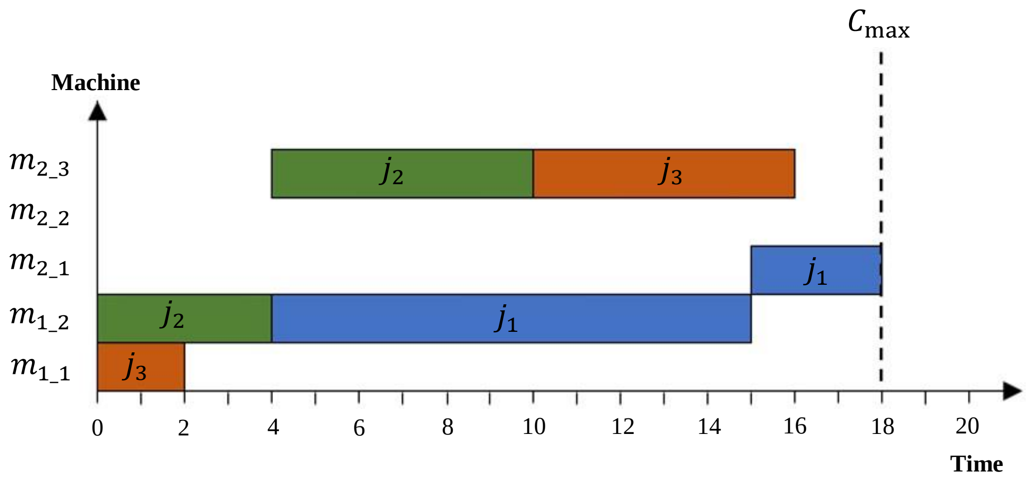

Figure 1 Example of generated production schedule

3. Tiki-Taka algorithm

Tiki-Taka Algorithm (TTA) is a population-based metaheuristic that inspired by football playing style [18]. Tiki-Taka is a well-known football playing style that refers to short passing, player positioning, and possession maintaining. The tiki-taka strategy is popularly associated with the Spanish national team and the football clubs in La Liga league. The initial objective of this tactic is to control the game by maintaining the ball possession. This is conducted by forming a triangle around the opponent players and making short passing. Once the triangle carries more than one opponent player, a new triangle will be formed. In order to form a triangle, players will always look for the key players in the team. When the ball is close to the opponent's penalty box, the key player will make some movement and strike to score a goal.

In Tiki-Taka Algorithm, the player position represents the candidate solution. Meanwhile, the ball position is the solution vector that will determine the position of the next player. Besides that, the key players represent a set of solution leaders that also influence the ball position in the game. The TTA implements the two main features from tiki-taka tactics: short passing and player positioning. The short passing feature aims to collect information from nearby player to decide the new player position. Other than the information of nearby player, the new player position is also obtained information of key player and the ball position. The flow of TTA is presented in Figure 2.

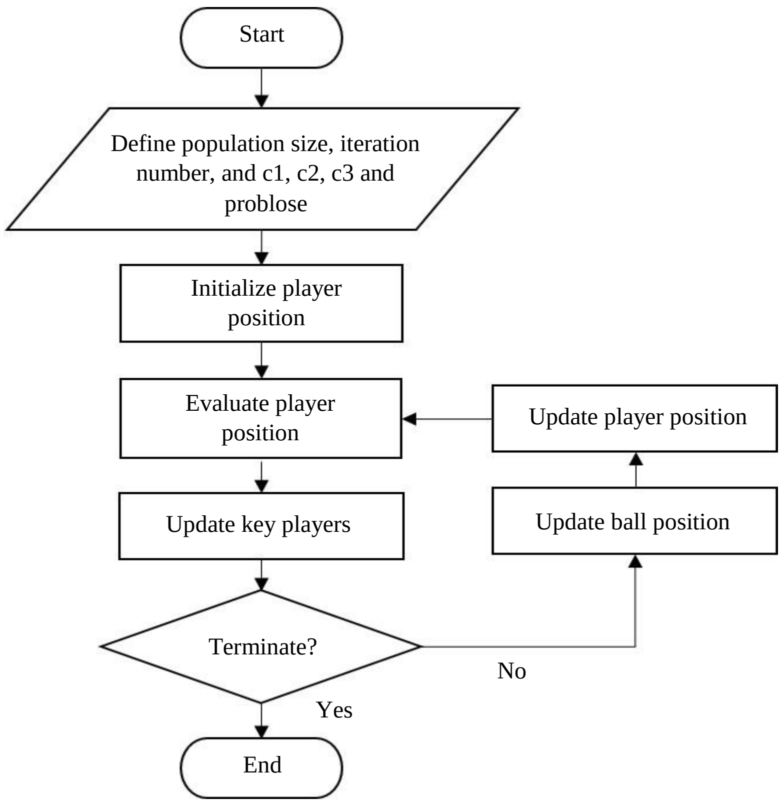

Figure 2 Flow chart of Tiki-Taka algorithm

3.1 Initialization

At the initial stage, the player position is randomly created. Before that, the parameters for the algorithm are defined, including the number of players, problem dimension, key players, and coefficients. Consider an HFS problem with n jobs and m machines, the problem is represented using $n \times m$ dimensions. A set of w players with $n \times m$ dimensions are randomly created known as P_w. This is the candidate solution with population size w.

$$P_w = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n\times m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n\times m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{w,1} & p_{w,2} & \cdots & p_{W,n\times m} \end{pmatrix}$$

Concurrently, another matrix that represents the ball position, B, is generated. At the initial stage, $B_w = P_w$.

$$B_w = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n\times m} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n\times m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{w,1} & b_{w,2} & \cdots & b_{W,n\times m} \end{pmatrix}$$

3.2 Evaluation

The evaluation is made to assess the player position according to optimization objective. Before the designated fitness function can be implemented, the player position must be decoded to transform it into an HFS solution. For this purpose, the player position, p_w, is divided into specific job size. For example, for problem with 3 jobs and 2 stages with two and three machines respectively, the dimension of solution will be 15. $Let\ P_w = [P_{w,1}, P_{w,1}, P_{w,1}, \dots, P_{w,15}]$. For decoding purposes, the p_w will be divided with n jobs, thus becoming X_i while the element inside the matrix are noted as $X_{i,s,k}$ Noted that the X_i and $X_{i,s,k}$ notations are only used during decoding and evaluation stages only.

$$X_i = \begin{bmatrix} x_{1,1,1} & x_{1,1,2} & x_{1,2,1} & x_{1,2,2} & x_{1,2,3} \\ x_{2,1,1} & x_{2,1,2} & x_{2,2,1} & x_{2,2,2} & x_{2,2,3} \\ x_{3,1,1} & x_{3,1,2} & x_{3,2,1} & x_{3,2,2} & x_{3,2,3} \end{bmatrix}$$

$X_{i,s,k}$ is the solution representation for job i at the kth machine on stage s. Noted that the X_i is randomly created for initial iteration, while for the rest of iterations, the value is derived from the TTA algorithm through new player position. In order to determine the job processing sequence, the minimum value from every row in X_i is identified as $X_{i_{min}}$ Then, $X_{i_{min}}$ is sorted in an ascending order known as X'_j. Finally, the job sequence J is derived from the job index, J. Similarly, the job-machine assignment is made based on the smallest representation for the job in each stage.

$$X_{i_{\min}} = \begin{bmatrix} x_{1_{\min}} \\ x_{2_{\min}} \\ x_{3_{\min}} \end{bmatrix}$$

$$X'_j = \text{ascending}\left\{X_{i_{\min}}\right\}$$

$$J = \{j\}; \quad \forall j = 1, 2, ..., n$$

The job processing sequence and job-machine assignment solutions are then evaluated using fitness function. Equations (1) and (2) present the objective functions to minimize completion time and total energy utilization. Since this problem is a multi-objective problem, one of the methods for handling the problem is using weighted sum approach. The weighted sum approach combines both optimization objectives into a single function. The single function is then optimized and treated as a single objective problem. However, since the C_{max} and TEC have a different range, they need to be normalized into a similar range [0, 1], as shown in equations (9) and (10). In order to identify the minimum and maximum values for each objective function ($C_{max_{max}}, C_{max_{min}}, TEC_{max}, TEC_{min}$), a thorough test using one objective function at a time is conducted. Finally, the fitness function, as in equation (11), is used in the optimization process. In equation (11), w_1 and w_2 are weightage $\hat{C}_{max}$ and $\widehat{TEC}$ respectively. In this work, the w_1 and w_2 are set to 0.5 each. This indicated that both objective functions are equally important.

$$\hat{C}_{\max} = \frac{C_{\max} - C_{\max_{\min}}}{C_{\max_{\max}} - C_{\max_{\min}}} \tag{9}$$

$$\widehat{TEC} = \frac{TEC - TEC_{\min}}{TEC_{\max} - TEC_{\min}} \tag{10}$$

$$\min f = w_1 \hat{C}_{\max} + w_2 \widehat{TEC} \tag{11}$$

3.3 Update key players

In TTA, key players are a set of best solutions from the population. The number of key players, n_k, is equivalent to 10% of the population size or a minimum of three players. The key players are selected and updated based on the top fitness values obtained from evaluation procedure. In HFS, the key players are the set of problem representations that came out with best C_{max} in the schedule. If the population size is 30, three key players (equal to 10%) will be selected based on the best fitness values. The key players procedure mimics the tiki-taka strategy that uses multiple key players in a team to diverse the ball movement. In optimization, this concept enables the algorithm to maintain the diversity of solution. The key player archive, h, will be updated in every iteration.

3.4 Update ball position

One of tiki-taka features is short passing. The ball updating procedure simulates this feature in order to deliver the ball from one player to the next nearby player, as shown in Figure 3. However, in a real play, there will be a possibility that the ball possession is lost to the opponent during the passing. In TTA, the probability of lose the ball ($prob_{lose}$) is set at 10%. The updated ball position, b_i', is formulated as follow. In this formula, r_p, is random number between 0 and 1.

$$b'_i = \begin{cases} rand\left(b_i - b_{i+1}\right) + b_i, & r_p > prob_{lose} \\ b_i - \left(c_1 + rand\right)\left(b_i - b_{i+1}\right), & r_p \le prob_{lose} \end{cases} \tag{12}$$

The ball updating formula in (12) consists of two conditions: successful pass ($r_p > prob_{lose}$) and unsuccessful pass ($r_p \le prob_{lose}$). For successful pass, the ball simply being delivered to the nearby player with some random. Meanwhile, for unsuccessful pass, the ball is being blocked. The c_1 is a coefficient that determines the ball reflection magnitude.

3.5 Update player position

The player position will be updated following the ball and also key player's positions. Since the TTA employed more than one leader, the key player selection is made randomly. Then, the player position is updated using the following formula:

$$p_i' = p_i + rand * c_2 * \left(b_i' - p_i\right) + rand * c_3 * \left(h - p_i\right) \tag{13}$$

In equation (13), c_2 and c_3 are coefficients ranging 1.5-2.5 and 0.5-1.5, respectively. These coefficients balanced the exploration and exploitation in the search space.

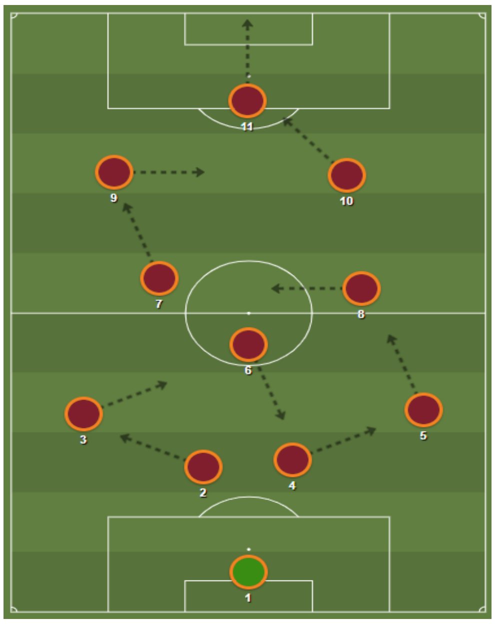

Figure 3 Tiki-Taka short passing example

3.6 Numerical example of TTA

In initial stage, the initial solution will be randomly created. For demonstration purpose, only four candidate solutions are showed. Besides the initial random solution, the TTA parameters also defined in this step. Let c_1 = 1.5, c_2 = 2.5, c_3 = 1.2, $prob_{lose}$ = 10% and initial solution as follow:

P_1	[0.581	0.592	0.368	0.312	0.220	0.746	0.547	0.703	0.819	0.532	0.752	0.342	0.447	0.775	0.083]
P_2	[0.021	0.477	0.598	0.389	0.301	0.444	0.893	0.655	0.633	0.047	0.969	0.265	0.030	0.025	0.993]
P_3	[0.359	0.284	0.185	0.653	0.024	0.595	0.901	0.262	0.002	0.929	0.668	0.276	0.769	0.031	0.097]
P_4	[0.415	0.670	0.926	0.292	0.177	0.843	0.202	0.360	0.894	0.628	0.161	0.617	0.144	0.592	0.804]

Then, the solutions are broken down into X_i, based on the job and machine number, thus becoming:

$$X_1 = \begin{bmatrix} 0.581 & 0.592 & 0.368 & 0.312 & 0.220 \\ 0.746 & 0.547 & 0.703 & 0.819 & 0.532 \\ 0.752 & 0.342 & 0.447 & 0.775 & 0.083 \end{bmatrix} \quad X_2 = \begin{bmatrix} 0.021 & 0.477 & 0.598 & 0.389 & 0.301 \\ 0.444 & 0.893 & 0.655 & 0.633 & 0.047 \\ 0.969 & 0.265 & 0.030 & 0.025 & 0.993 \end{bmatrix}$$

$$X_3 = \begin{bmatrix} 0.359 & 0.284 & 0.185 & 0.653 & 0.024 \\ 0.595 & 0.901 & 0.262 & 0.002 & 0.929 \\ 0.668 & 0.276 & 0.769 & 0.031 & 0.097 \end{bmatrix} \quad X_4 = \begin{bmatrix} 0.415 & 0.670 & 0.926 & 0.292 & 0.177 \\ 0.843 & 0.202 & 0.360 & 0.894 & 0.628 \\ 0.161 & 0.617 & 0.144 & 0.592 & 0.804 \end{bmatrix}$$

The candidate solution is decoded solution using the approach mentioned in section 2.1.1. The decoded solution later is evaluated using equation (11). The obtained fitness value for X_1 until X_4 are as follow;

X_i	$f(X_i)$
X_1	0.5421
X_2	0.8634
X_3	0.7415
X_4	0.3766

Now the key players are updated by selecting the top solutions based on the fitness value. For demonstration purpose, let assume the n_k is 2. Based on the fitness value, $f(X_i)$, the top two solutions were X_1 and X_4. Since X_1 and X_4 were coming from P_1 and P_4, the key players, h will consist of P_1 and P_4. Noted that in actual application, number of key players will be 10% of the population size.

$$h = \{P_1, P_4\}$$

Next, the ball position is updated. For initial solution, $B_w = P_w$. In this step, the calculation using equation (12) is conducted dimension by dimension. In this example, only the first dimension of solution 1 ($P_{1,1}$) will be demonstrated. In order to conduct this operation, a random number is generated. Lets the random number for $P_{1,1}$ is 0.3522. Since the random number is larger than $prob_{lose}$ (0.1), the updated ball position, will use the following equation (from equation (12)).

$$b'_{1,1} = rand * (b_{1,1} - b_{2,1}) + b_{1,1}$$

Replacing the values into this equation,

$$b'_{1,1} = 0.3522(0.581 - 0.021) + 0.581$$
$$b'_{1,1} = 0.778$$

Then, the player position is updated using equation (13). In this step, the algorithm will randomly select one key player from h. Lets the selected key player is P_4. Therefore, the h value in equation (13) will be replaced with $p_{4,1}$.

$$p'_{1,1} = p_{1,1} + rand * c_2 * (b'_{1,1} - p_{1,1}) + rand * c_3 * (p_{4,1} - p_{1,1})$$

Lets random numbers are 0.7942 and 0.0660, replacing the values;

$$p'_{1,1} = 0.581 + 0.7942 * 2.5 * (0.778 - 0.581) + 0.0660 * 1.2 * (0.415 - 0.581)$$
$$p'_{1,1} = 0.959$$

The procedures above is repeated for all solution dimensions. Then a new player position, P'_1 will be established. This procedure is also repeated for other candidate solutions.

4. Results and discussion

This section presents the results obtained from computational experiment and case study problem. The results were then thoroughly discussed to relate the findings with theoretical in optimization and scheduling. The purpose of computational experiment is to verify the proposed TTA performance in optimizing the HFS with energy consumption problem. Meanwhile the case study problem aims to validate the applicability of the proposed model and algorithm in the real-life problem.

4.1 Computational experiment

A computational experiment has been conducted to test the proposed TTA to optimize HFS with energy utilization. For this purpose, a set of benchmark test problem proposed by Carlier and Neron [19] was utilized. This is one of popular hypothetical test problem that widely used to test hybrid flow shop scheduling problem. Table 6 demonstrates the details of the benchmark test problem. The processing time is randomly generated using normal distribution between {3, 20}. The last column in Table 6 indicated the machine configuration at each stage. In j10c5a2, for instance, there are two machines in each stage, except in the third stage that had only one machine.

Table 6 Benchmark test problem configuration

Problem	Number of jobs	Number of stages	Machine configuration
j10c5a2	10	5	2 2 1 2 2
j10c5b1	10	5	1 2 2 2 2
j10c5c1	10	5	3 3 2 3 3
j10c5d1	10	5	3 3 3 3 3
j10c10a2	10	10	2 2 2 2 1 2 2 2 2 2
j10c10b1	10	10	1 2 2 2 2 2 2 2 2 2
j10c10c1	10	10	3 3 3 3 2 3 3 3 3 3
j15c5a1	15	5	3 3 1 3 3
j15c5b1	15	5	1 3 3 3 3
j15c5c1	15	5	3 3 2 3 3
j15c10a2	15	10	3 3 3 3 1 3 3 3 3 3
j15c10b1	15	10	1 3 3 3 3 3 3 3 3 3

To compare the performance of TTA, six other metaheuristic algorithms were used in the optimization process. The comparison algorithms were Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Firefly Algorithm (FA), Moth Flame Optimizer (MFO), and Butterfly Optimization Algorithm (BOA). These comparison algorithms were chosen based on different factors. ACO and PSO were chosen because of their popularity in metaheuristics in optimizing the combinatorial problem. In addition, these metaheuristics were among the well-established algorithms introduced in the 1990s. On the other hand, ABC and FA were introduced in 2005 and 2008, respectively. They represented the popular mid-age metaheuristics that were implemented to optimize various type of combinatorial problems. Finally, the MFO and BOA represented the recent metaheuristics algorithms. The MFO was introduced in 2015, while BOA in 2018.

In the computational experiment, the population size was set to 30, while the maximum iteration was 1,000 for all algorithms. The optimization run was repeated 20 times for each of benchmark problem. The specific parameter setting for algorithms is presented in Table 7. Noted that for TTA, the parameter setting utilized in this work is based on previous study in literature [18].

Table 7 Parameter setting for TTA and comparison algorithms

Algorithm	Parameter setting
ACO [20]	Relative importance of pheromone: α = 1 Relative importance of heuristic: β = 3 Local evaporation rate: 0.1 Global evaporation rate: 0.2
PSO [21]	Inertia coefficient, w = 1 Cognition learning coefficient, c_1 = 1.8 Social learning coefficient, c_2 = 1.8
ABC [22]	Crossover probability = 0.6 Local search rate = 0.4 Scout Limit = 1/8
FA [23]	Attraction coefficient = 2 Light absorption coefficient = 1 Randomization parameter = 0.5
MFO [24]	Constant to interpret the shape of logarithmic spiral, b = 1 Closeness to the flame, t: linearly decrease from 1 to -1
BOA [25]	Modular modality c: 0.01 Power exponent a: increased from 0.1 to 0.3 Switch probability p: 0.8
TTA [18]	Probability of lose the ball ($prob_{lose}$) : 10% Ball reflection magnitude coefficient, c_1 = 1.5 Ball position coefficient, c_2 = 2.5 Key player’s position coefficient, c_3 =1.2

Table 8 presents the mean and standard deviation of fitness value obtained from computational experiment. In general, the TTA came out with the best mean fitness in nine out of twelve benchmark test problems. Meanwhile, in the remaining three problems, TTA ranked second. According to Table 8, the next best algorithm was FA. The FA obtained the best mean in one problem, ranked second in nine problems and fourth in two problems. The results indicated that the TTA was able to converge consistently for every optimization run.

Table 8 Mean and standard deviation of fitness function from computational experiment

Problem	Indicator	ACO	PSO	ABC	FA	MFO	BOA	TTA
j10c5a2	Mean	0.2475	0.3012	0.2401	0.2561	0.2780	0.3427	0.2417
	SD	0.0084	0.0221	0.0054	0.0060	0.0126	0.0197	0.0062
j10c5b1	Mean	0.2807	0.3432	0.2765	0.2888	0.3027	0.3403	0.2791
	SD	0.0056	0.0165	0.0029	0.0075	0.0138	0.0261	0.0071
j10c5c1	Mean	0.2035	0.2461	0.1890	0.1713	0.2208	0.2996	0.1592
	SD	0.0073	0.0232	0.0098	0.0131	0.0215	0.0312	0.0164
j10c5d1	Mean	0.3183	0.3481	0.3086	0.2945	0.3039	0.3921	0.2908
	SD	0.0094	0.0187	0.0069	0.0139	0.0196	0.0278	0.0146
j10c10a2	Mean	0.3681	0.4045	0.3553	0.3419	0.3966	0.4864	0.3293
	SD	0.0107	0.0310	0.0034	0.0150	0.0157	0.0237	0.0125
j10c10b1	Mean	0.3802	0.4223	0.3668	0.3435	0.3953	0.4821	0.3193
	SD	0.0110	0.0208	0.0111	0.0236	0.0210	0.0280	0.0110
j10c10c1	Mean	0.3033	0.3330	0.2944	0.2419	0.2883	0.4842	0.2319
	SD	0.0169	0.0251	0.0133	0.0267	0.0313	0.0171	0.0225
j15c5a1	Mean	0.4079	0.4533	0.3947	0.3917	0.4138	0.4948	0.3878
	SD	0.0094	0.0293	0.0046	0.0103	0.0118	0.0204	0.0065
j15c5b1	Mean	0.4939	0.5306	0.4820	0.4746	0.4949	0.5888	0.4744
	SD	0.0052	0.0243	0.0084	0.0100	0.0101	0.0233	0.0054
j15c5c1	Mean	0.2946	0.3037	0.2774	0.2326	0.2723	0.4955	0.2305
	SD	0.0127	0.0158	0.0114	0.0075	0.0183	0.0157	0.0130
j15c10a2	Mean	0.2987	0.3279	0.2819	0.2481	0.2958	0.4968	0.2564
	SD	0.0148	0.0204	0.0112	0.0114	0.0231	0.0249	0.0184
j15c10b1	Mean	0.2709	0.3182	0.2595	0.2412	0.2894	0.3942	0.2206
	SD	0.0190	0.0361	0.0085	0.0072	0.0147	0.0283	0.0155

In order to test the significance of TTA results, a nonparametric test was conducted using Wilcoxon signed-rank test at 95% confidence interval. The nonparametric test was chosen because the results obtained from optimization did not normally distributed. Table 9 indicates the p-value from the Wilcoxon signed-rank test. When the p-value is smaller than significance level (0.05), it indicated that there is significant difference between TTA and comparison algorithm results. The p-value with asterisk (*) symbol in Table 9 shows that the TTA has no significant difference compared with comparison algorithm.

In general, TTA has a significant difference in all benchmark problems compared with PSO and BOA. Meanwhile, in comparison with ACO and MFO, TTA is significantly better in 91.7% of the benchmark problems. However, TTA is only significantly better than FA in 16.7% (or two benchmark problems) of the problems. In summary, TTA has a significant different in 80.6% of the total cases in the computational experiment problems.

Table 9 Wilcoxon signed-rank test (p-value)

Problem	ACO	PSO	ABC	FA	MFO	BOA
j10c5a2	0.0403	0.0002	0.5933*	0.0207	0.0002	0.0002
j10c5b1	0.3060*	0.0002	0.5946*	0.2378*	0.0004	0.0002
j10c5c1	0.0002	0.0002	0.0008	0.2413*	0.0002	0.0002
j10c5d1	0.0008	0.0002	0.0091	0.4727*	0.1212*	0.0002
j10c10a2	0.0002	0.0002	0.0013	0.1212*	0.0002	0.0002
j10c10b1	0.0002	0.0002	0.0002	0.0073	0.0002	0.0002
j10c10c1	0.0002	0.0002	0.0002	0.3447*	0.001	0.0002
j15c5a1	0.0010	0.0002	0.0257	0.1859*	0.0002	0.0002
j15c5b1	0.0028	0.0002	0.0452	0.6776*	0.0036	0.0452
j15c5c1	0.0002	0.0002	0.0002	0.9698*	0.0004	0.0002
j15c10a2	0.0002	0.0002	0.0028	0.3075*	0.0028	0.0002
j15c10b1	0.0008	0.0002	0.0022	0.0539*	0.0022	0.0002

4.2 Case study

In order to demonstrate the applicability of the proposed HFS with energy utilization model and algorithm in real-life problems, an industrial case study was conducted in a bearing machining process. There were three main processes for bearing outer ring: face grinding, groove grinding, and bore grinding. In this study, the number of machines was limited to three machines with different capacity at each stage. There were six jobs that needed to be scheduled for a one-month period. The job processing time at each machine is presented in Table 10. The last row in Table 10 indicates the power rating for each machine.

Table 10 Processing time for bearing machining process (minutes)

Job	Quantity	Facing			Grooving			Boring		
		MF1	MF2	MF3	MG1	MG2	MG3	MB1	MB2	MB3
J1	1200	864	624	576	1440	2040	1920	1008	1008	864
J2	3750	2700	1950	1800	4500	6375	6000	3150	3150	2700
J3	3250	2340	1690	1560	3900	5525	5200	2730	2730	2340
J4	2500	1800	1300	1200	3000	4250	4000	2100	2100	1800
J5	4150	2988	2158	1992	4980	7055	6640	3486	3486	2988
J6	2250	1620	1170	1080	2700	3825	3600	1890	1890	1620
Power (W)		2800	3200	3600	4200	3800	3200	1800	1800	2400

Optimization for the case study problem was conducted using TTA, FA, ABC, and MFO. The optimization run was repeated 20 times; the average and best fitness were recorded. Table 11 presents the optimization result for the case study problem. The proposed TTA had consistently performed better in average and best fitness, compared with FA, ABC, and MFO. The last two columns in Table 11 demonstrate the individual optimization objective that was decoded from the obtained best fitness solution. From the result, the minimum C_{max} obtained by TTA was 12,760 minutes. Meanwhile, the minimum TEC was established by MFO. Nevertheless, the C_{max} for MFO was the highest among all algorithms. In comparison with other algorithms, C_{max} for TTA has a better performance between 0.3%-2.0%. However, the TEC for TTA was 0.18% and 0.25% higher than FA and MFO, respectively. The results indicated that none of the optimization algorithms were able to find minimum values in both optimization objectives. The TTA result was compromised between the C_{max} and TEC values.

Table 11 Optimization result of case study problem

Algorithm	Average Fitness	Best Fitness	C_{max} (minute)	TEC (kWh)
TTA	0.4725	0.4516	12760	2466.5
FA	0.4741	0.4527	12819	2461.9
ABC	0.4783	0.4576	12802	2472.3
MFO	0.4780	0.4649	13020	2460.4

Figure 4 presents the convergence for TTA, FA, ABC, and MFO in solving case study problem. The FA converged at the initial stage and maintained the obtained fitness. The MFO also converged at early stage and at the middle of the iterations. In contrast, the ABC algorithm converged at the early stage, then maintained the convergence at the middle stage and experienced convergence again at the end of iteration. The proposed TTA convergence can be observed in the early and middle iteration process. The last final convergence occurred approximately near the 700th iteration. This result indicated the ability of TTA to converge until two-thirds of iteration. The flat curve after 700th iteration confirmed the attainment of optimum solution.

Figure 5 presents the production schedule obtained by TTA that was presented in Table 9. The production begins by processing J6, J5, and J2 on MF1, MF2, and MF3, respectively. There was a minimum utilization of machine with higher power rate, especially MF3 and MB3 in the schedule, to reduce TEC. In MB2, there were gaps between J6-J2 and J2-J1. This situation was because of the precedence constraints of both jobs in the second stage (Grooving). However, in the case that the completion time is not critical, the production planner might select solution from FA or MFO that provides lower energy consumption.

Numerical experiment and case study problem optimization clearly indicated that TTA provided better solution compared with comparison algorithms. There are a few factors that contributed to exceptional TTA performance compared with existing algorithms. The first unique feature is short passing strategy to the nearby solution. This feature allows the algorithm to collect information from other solutions before deciding the new solution. It also means that all candidate solutions will contribute to the new solution reproduction. From this strategy, the algorithm will identify suitable/unsuitable position to move.

The second feature of TTA is the multiple key players (leader solutions) in the algorithm. Majority of existing metaheuristics use single leader solution to influence the reproduction of new solutions towards better solution. Using this strategy, the candidate solution tends to move nearby the leader. If the leader solution was in local optima, all the solution having risk to be trapped. Meanwhile in TTA, each of candidate solutions need to randomly choose a key player to be considered from the archive during updating player position. This feature makes the candidate solutions more diverse and also reduce possibility to be trapped in local optima. Both of the unique features make TTA performed better than other algorithms in this work.

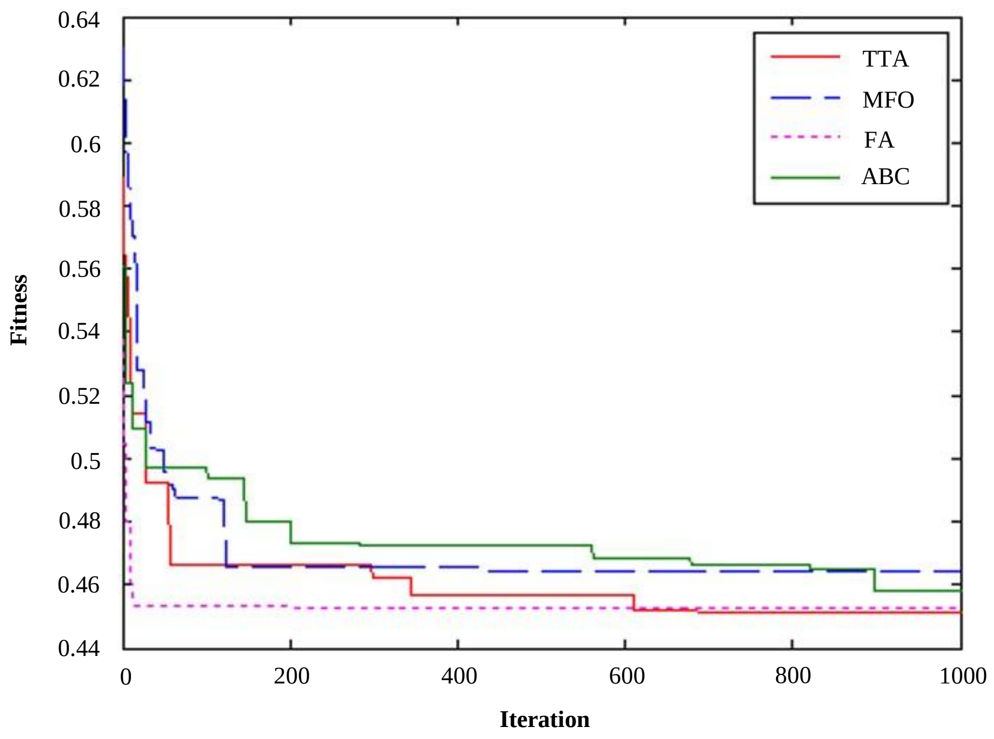

Figure 4 Convergence plot for case study problem

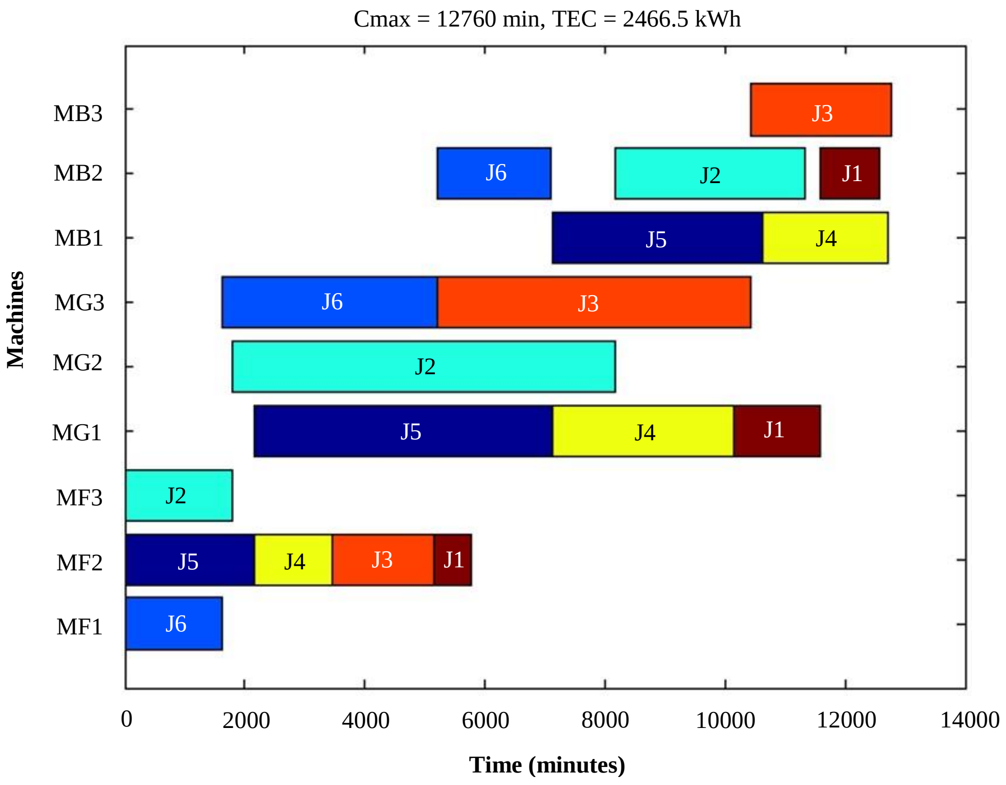

Figure 5 Production schedule using TTA

5. Conclusions

This paper presented hybrid flow shop scheduling (HFS) problem optimization using a new algorithm called Tiki-Taka Algorithm (TTA). In difference with the regular flow shop scheduling, the studied problem considered the total energy consumption as one of optimization objectives towards sustainable manufacturing systems. The research begins with formulation of hybrid flow shop scheduling with energy consumption. Then, a computational experiment was conducted to assess TTA performance for HFS with energy consumption using a set of benchmark problems. The results were compared with well-established, mid-age, and recent optimization algorithms. Then, a case study problem is presented to demonstrate applicability of the proposed HFS with energy consumption and TTA in optimizing real-life problems.

The computational experiment results indicated that TTA performed best in 75% of the benchmark test problems, and second best in the remaining problems. Based on statistical test, the TTA solution has significant difference in 80.6% of the cases compared with other algorithms. The results show the superiority of TTA to optimize HFS with energy consumption. On the other hand, the case study optimization confirmed the computational experiment result. The TTA showed minimum average and best fitness compared with comparison algorithm. The obtained optimization objective values by TTA had compromised the total makespan and energy consumption.

6. Acknowledgement

The authors would like to acknowledge Universiti Malaysia Pahang for funding this research under the internal grant RDU190317 (Energy Utilization Modelling and Optimization for Machine Scheduling Problem in Production).

7. References

[1] Shao W, Shao Z, Pi D. Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem. Knowl Base Syst. 2020;194:105527.

[2] Fernandez-Viagas V, Perez-Gonzalez P, Framinan JM. Efficiency of the solution representations for the hybrid flow shop scheduling problem with makespan objective. Comput Oper Res. 2019;109:77-88.

[3] Tosun O, Marichelvam MK, Tosun N. A literature review on hybrid flow shop scheduling. Int J Adv Oper Manag. 2020;12(2):156-94.

[4] Li J, Sang H, Han Y, Wang C, Gao K. Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. J Clean Prod. 2018;181(3):584-98.

[5] Schulz S. A genetic algorithm to solve the hybrid flow shop scheduling problem with subcontracting options and energy cost consideration. In: Wilimowska Z, Borzemski L, Swiątek J, editors. Proceedings of 39th International Conference on Information Systems Architecture and Technology-ISAT; 2018 Sep 16-18; Nysa, Poland. Berlin: Springer; 2019. p. 263-73.

[6] Li M, Lei D, Cai J. Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. Swarm Evol Comput. 2019;49:34-43.

[7] Schulz S, Neufeld JS, Buscher U. A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling. J Clean Prod. 2019;224:421-34.

[8] Geng K, Ye C, Dai ZH, Liu L. Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs. Complexity. 2020;2020:8565921.

[9] Jiang S, Zhang L. Energy-oriented scheduling for hybrid flow shop with limited buffers through efficient multi-objective optimization. IEEE Access. 2019;7:34477-87.

[10] Liu Z, Yan J, Cheng Q, Yang C, Sun S, Xue D. The mixed production mode considering continuous and intermittent processing for an energy-efficient hybrid flow shop scheduling. J Clean Prod. 2020;246:119071.

[11] Gong G, Chiong R, Deng Q, Han W, Like Z, Lin W, et al. Energy-efficient flexible flow shop scheduling with worker flexibility. Expert Syst Appl. 2020;141:112902.

[12] Meng L, Zhang C, Shao X, Ren Y, Ren C. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int J Prod Res. 2019;57(4):1119-45.

[13] Zhou R, Lei D, Zhou X. Multi-objective energy-efficient interval scheduling in hybrid flow shop using imperialist competitive algorithm. IEEE Access. 2017;7:85029-41.

[14] Tao X, Li J, Huang T, Duan P. Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption. Complex Intell Syst. 2020;7(1):1-16.

[15] Utama DM, Widodo DS, Ibrahim MF, Hidayat K, Baroto T, Yurifah A. The hybrid whale optimization algorithm: a new metaheuristic algorithm for energy-efficient on flow shop with dependent sequence setup. J Phys Conf Ser. 2020;1569:22094.

[16] Geng K, Ye C, Cao L, Liu L. Multi-objective reentrant hybrid flowshop scheduling with machines turning on and off control strategy using improved multi-verse optimizer algorithm. Math Probl Eng. 2019;2019:1-18.

[17] Utama DM, Baroto T, Widodo DS. Energy-efficient flow shop scheduling using hybrid grasshopper algorithm optimization. J Ilmiah Teknik Industri. 2020;19(1):30-8.

[18] Rashid M. Tiki-Taka algorithm: a novel metaheuristic inspired by football playing style. Eng Comput. 2021;38(1):313-43.

[19] Carlier J, Neron E. An exact method for solving the multi-processor flow-shop. RAIRO Oper Res. 2000:34(1):1-25.

[20] Luo H, Du B, Huang GQ, Chen H, Li X. Hybrid flow shop scheduling considering machine electricity consumption cost. Int J Prod Econ. 2013;146(2):423-39.

[21] Tang D, Dai M, Salido MA, Giret A. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. Comput Ind. 2016;81:82-96.

[22] Gong D, Han Y, Sun J. A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. Knowl Base Syst. 2018;148:115-30.

[23] Rashid MF, Osman MA. Optimisation of energy efficient hybrid flowshop scheduling problem using firefly algorithm. 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE); 2020 Apr 18-19; Malaysia. New York: IEEE; 2020. p. 36-41.

[24] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Base Syst. 2015;89:228-49.

[25] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput. 2019;23(3):715-34.