INTERNATIONAL JOURNAL OF AUTOMOTIVE AND MECHANICAL ENGINEERING
ISSN: 2229-8649 e-ISSN: 2180-1606
VOL. 21, ISSUE 3, 2024, 11616 – 11628
DOI: https://doi.org/10.15282/ijame.21.3.2024.13.0896

RESEARCH ARTICLE

Optimization of Cost-Based Hybrid Flowshop Scheduling Using Teaching-Learning-Based Optimization Algorithm

W.Ullah[1], M.A.N. Mu'tasim[2], M.F.F. Rashid[2]*

[1]Faculty of Manufacturing and Mechatronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pahang, Malaysia
[2]Faculty of Mechanical and Automotive Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pahang, Malaysia

ABSTRACT - A cost-based hybrid flowshop scheduling (CHFS) combines flow shop and job shop elements, with cost considerations as a key indicator. CHFS is a complex combinatorial optimization challenge encountered in real-world manufacturing and production environments. This paper investigates the optimization of a CHFS problem using the Teaching Learning-Based Optimization (TLBO) algorithm. Effective CHFS is crucial for achieving production balance, reducing costs, and improving customer satisfaction. The authors formulate the CHFS scheduling problem and propose applying the TLBO algorithm to minimize total costs, including labor, energy, maintenance, and delay expenses. The performance of the TLBO technique is evaluated through computational experiments on various CHFS problem instances. The results demonstrate the effectiveness of the TLBO algorithm, which achieved the best results in 42% of the test cases, surpassing other algorithms like the Grey Wolf Optimizer and Particle Swarm Optimization. Additionally, the TLBO algorithm had the highest average performance ranking across the comparative algorithms. The study highlights the potential of the TLBO algorithm as an efficient optimization tool for complex manufacturing scheduling problems.

ARTICLE HISTORY

Received	:	25th May 2024
Revised	:	18th July 2024
Accepted	:	01st Aug. 2024
Published	:	20th Sept. 2024

KEYWORDS
Cost optimization
Hybrid flowshop
Scheduling
TLBO algorithm

1. INTRODUCTION

The hybrid flow shop scheduling (HFS) problem has been extensively studied due to its significance in manufacturing and its impact on production systems. The HFS is an intricate combinatorial problem involved in many real-world scenarios [1]. The study of HFS problems arises from the need to optimize production processes and enhance overall operational efficiency. Efficient scheduling in a hybrid flowshop environment can lead to balanced production, cost reduction, improved customer satisfaction, and a competitive edge in the manufacturing and production industries.

Cost optimization in manufacturing processes is of paramount importance [2]. Cost reduction helps manufacturers minimize related expenses such as labor, energy, material, and overhead, leading to improved profitability in the market. Cost optimization in the manufacturing process is crucial for profitability, resource efficiency, productivity, competitiveness, budget control, sustainability, and adaptability in a dynamic marketplace [3]. Furthermore, cost optimization is vital for staying competitive, embracing lean principles, driving continuous improvement, managing risks, supporting investment and growth, optimizing the supply chain, complying with regulations, and delivering value to customers. It is a multifaceted approach that permeates all aspects of a manufacturing organization and contributes to its overall success.

There are plenty of scheduling algorithms that have been used by many researchers [4]. In the beginning, researchers used the heuristic approach to optimize scheduling problems. The heuristics algorithms include first come first serve (FCFS), earliest due date (EDD), shortest processing time (SPT) and longest processing time (LPT) [5]. On top of these heuristics techniques, practitioners also developed advanced optimization tools and techniques known as metaheuristics. Metaheuristics are high-level optimization tools that are capable of providing near-optimal solutions [6]. The well-established metaheuristic algorithms for optimization problems include Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA) [7], [8], [9]. Besides these algorithms, other algorithms such as Differential Evolution (DE) and Harmony search (HS) have also been explored for HFS scheduling optimization problems [10], [11]. However, these scheduling algorithms have limitations in terms of problem type and size. It should be noted that each of the metaheuristic algorithms has a different working mechanism and approach to solve optimization problems. Each algorithm has limited capacity to tackle the scheduling problem technicality and adapt itself according to problem size and complexity. There is a need for efficient algorithms to solve complex permutation problems if the problem size and dimension vary.

Teaching learning-based optimization (TLBO) algorithm is a powerful population-based algorithm introduced by Rao in 2011 [12]. Since then, this algorithm has been employed in multiple fields, such as chemistry, engineering, computer science and economics. The TLBO algorithm works based on the teaching and learning process in a classroom. In literature, this algorithm has shown promising results for scheduling problems. Its potential for scheduling problems lies

***CORRESPONDING AUTHOR | M.F.F. Rashid |** ✉ ffaisae@umpsa.edu.my

in its ability to navigate the search space efficiently. By mimicking the teaching and learning process, TLBO can guide the search towards promising solutions while allowing for exploration to avoid getting trapped in local optima.

The rest of the paper is structured in the following manner: Section 2 reported the related works on HFS scheduling and optimization techniques. Section 3 formulates the cost-based hybrid flowshop (CHFS) scheduling problem, followed by the proposed TBLO algorithm in section 4. The computational experiment setup was organized in section 5. Section 6 displays the computational experiment results. The conclusion of the paper can be seen in section 7.

2. RELATED WORKS

In CHFS, most of the prior work encountered the makespan with energy cost optimization. In one paper, the energy cost optimization problem was addressed with unrelated parallel machine (UPM) scheduling [13]. In this scenario, each job goes through *k* stages, where each stage consists of m_s UPM. The advantage of UPM is its increased processing capacity due to the availability of multiple machines for each operation. Besides this, the total production cost, including maintenance and emergency repair costs, was used as a fitness function utilizing the distributed HFS (DHFS) [14]. Some other published work focuses on the optimization of the total cost, including raw material, vehicle, production, delivery, and tardiness [15].

As claimed earlier, energy cost optimization is the topmost objective function considered by researchers in their work. Regarding the energy-efficient HFS (EHFS), it was utilized in steel-making plants for energy cost saving [16]. The advantages of employing EHFS include reduced energy consumption, improved production efficiency, and enhanced resource utilization. The production cost is the second most frequently considered cost in CHFS problems. In one article, the production cost captured the cost of production equipment, tools, storage, scrap, materials, energy, and late delivery [2]. In some studies, production cost refers to transportation, labor, and maintenance costs [9], [17]. On top of these, penalty costs were also optimized, which is associated with the late delivery of jobs [18].

Regarding the optimization techniques for the CHFS problems, various metaheuristics were implemented. Among the popular metaheuristic algorithms used for optimization problems are Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE). For instance, in a paper published in 2019, GA was used to reduce total production costs, including transportation and energy costs [9]. Modified versions of GA, such as multi-layer encoding GA (MLEGA) and hybrid GA (HGA), have been employed to optimize production scheduling and reduce overtime costs, respectively [19], [20]. GA has been chosen for its performance, flexibility, and robustness in CHFS optimization. Improved GA versions, such as (IGA) with local search (LS), have shown effectiveness in solving CHFS problems quickly [21]. Other algorithms, such as Q-learning and general variable neighborhood search-driven non-dominated sorting GA (QVNS-NSGA-II), have also been used to optimize various costs in CHFS [22].

The PSO was used to optimize production costs, including maintenance and energy [3]. Modified versions of PSO, such as the hybrid PSO (HPSO) with Heuristics or Genetic algorithms, have been developed to improve solution quality and convergence speed [23]. PSO has been chosen for its simplicity, fast convergence, and ability to handle complex optimization problems. The discrete version of PSO (DPSO) has shown consistent and superior performance in various scenarios [8]. PSO can also be hybridized with other algorithms, such as Variable Neighborhood Search (VNS) or Improved Spider Monkey Optimization (I-SMO), to achieve more effective and efficient results [24], [25]. It serves as a base algorithm for comparative analysis and evaluation in optimization problems.

Researchers have applied the Differential evolution (DE) algorithm in various studies of CHFS to optimize production scheduling, processing costs, load balancing costs, delay costs, and energy costs [26]. Modified versions of DE, such as double population self-adaptive differential evolution (DPSADE) and bi-objective differential evolution (BODE), have been employed to enhance its performance in specific contexts [26]. DE has demonstrated effectiveness in solving CHFS problems and providing better results compared to similar metaheuristic algorithms. It offers advantages such as effective production cost optimization and superior solutions for load-balancing problems. The combination of DE with local search LS or other techniques further improves its optimization capabilities in CHFS.

The TLBO algorithm has attracted many researchers as it has great potential to optimize CHFS scheduling problems. Normally, the improved versions of TLBO were utilized for CHFS problem optimization. In one paper, the makespan and energy cost were optimized using the Multi-objective TLBO in HFS scheduling [27]. A study was conducted to investigate an EHFS problem involving additional resources and asymmetric machines. The optimization problem sought to simultaneously minimize the makespan and the total energy cost. To tackle this problem, the researchers propose an adaptive two-class teaching-learning-based optimization (ATLBO) algorithm that incorporates multiple teachers. The ATLBO algorithm is designed to find optimal schedules that minimize both the makespan and consumption of energy in a complex manufacturing environment [28]. Furthermore, a modified TLBO (*mTLBO*) algorithm was used to solve the re-entrant HFS scheduling problem (RHFSP) with an objective function of total tardiness and makespan [29]. In the green manufacturing era, reducing the consumption of energy should also be fully encountered in HFS scheduling problems. For this purpose, researchers propose a novel teachers' teaching-learning-based optimization (TTLBO) algorithm, which aims to optimize the total energy cost and the overall tardiness simultaneously [30].

Extensive research has been carried out to optimize various cost functions in the context of HFS scheduling problems. To the best of the author’s knowledge, none of the studies focus on optimizing holistic cost models in HFS scheduling using the standard TLBO algorithm. The contribution to this work is as follows: (a) A holistic cost model capturing four cost components, labor, electricity, maintenance, and late penalty cost simultaneously as a single optimization objective was established. (b) Conducting a comparative analysis of the TLBO algorithm through computational experiments with other metaheuristics to optimize the total production cost.

2.1 Problem Formulation

2.1.1 Problem description

The HFS scheduling problem with non-identical machines involves *N* jobs that must be processed through *S* stages in a predetermined order, with multiple parallel machines available at each stage. The two key decision variables are determining the job processing sequence at each stage and assigning jobs to specific machines. These decisions can significantly impact the makespan and total production cost, which have a tradeoff relationship. To address this challenge, the authors made certain assumptions in their investigation to optimize the CHFS scheduling problem using the TLBO algorithm, such as:

- No pre-emptive actions are allowed; current operations cannot be interrupted by new tasks.
- The data of problem are assured.
- All machines are available at initial time.
- An infinite buffer exists between two processes.
- Each machine can handle only one job at a time.
- Each stage allows only one machine to process each job.
- Each job must follow the specified process order without deviation.
- Unnecessary processes can be bypassed.
- Setup and transportation times are encountered in the overall processing time.
- All orders are finalized before the scheduling period begins.

2.1.2 Notations

For a better understanding of the problem, some notations were defined:

Indexes:

J: Represents the index of jobs, with $j = 1, 2, ..., n$.
s: Represents the index of stages, with $s = 1, 2, ..., n$.
l: Represents the index of jobs on a single machine, with $0 \leq l \leq n$.
m: Represents the index of machines at each stage, with $m = 1, 2, ..., m_j$.

Parameters:

J: Overall jobs
S: Total stages
Ms: Number of machines exists on each stage *s*
tj,s.m: Process time of job *j* on machine *m* at stage *s* (minute)
Cj: Finish time of job *j* (minute)
Dj: Due date of job *j* (minutes)
Cmax: Makespan (in minutes)
psm: power rating for machine *m* at stage *s* (Watt)
rsm: Periodical maintenance cost for machine *m* at stage *s* (in MYR)
trsm: Maximum operating duration before maintenance for machine *m* at stage *s* (in minutes)

2.1.3 Mathematical modeling

This paper aims to optimize production costs in hybrid flowshop (HFS) scheduling. Four cost factors were considered in determining the total production cost as the optimization objective. These factors are labor cost, energy cost, maintenance cost, and delay costs. Each of these costs is formulated as follows:

Labor cost (C_L) is the payment made to workers for their services in performing a particular job within a defined period. In this study, C_L is considered constant and is based on an hourly pay rate. It is assumed that each machine requires one worker to operate it. The labor cost C_L can be obtained by multiplying the total time of operation of all machines with the hourly wages. The time of machine operation is typically measured in minutes. Here, *J* represents overall jobs, *S* represents all stages, and *M* represents the total number of machines. The variable *tjsm* denotes the total machine processing time in minutes. Additionally, *αjsm* is a binary variable subject to certain constraints, as expressed in equation (1).

$$C_L = \left(\sum_{j=1}^{J}\sum_{s=1}^{S}\sum_{m=1}^{M} t_{jsm}\cdot\alpha_{jsm}\right)\times\left(\frac{Hourly\ payrate}{60}\right) \quad (1)$$

$$\alpha_{jsm} = \begin{cases} 1, & \text{if job } j \text{ is processed on machine } m \text{ at stage } s \\ 0, & \text{otherwise} \end{cases}$$

The electricity cost, denoted as (C_E), specifically refers to the electricity consumed by the machines during operating duration. Other electrical charges, such as ventilation and lighting, were excluded, as well as the power consumed by machines in standby mode or idle state. The power consumption of each machine is considered when calculating C_E. The power rating of each machine is multiplied by its operating duration, and the result is converted into kilowatt-hours (kWh). This study deals with non-identical machines, meaning that machines within the same stage may have different power ratings, which affects overall power usage. In equation (2), the first term is the total consumption of energy in watt-minutes. To convert into kWh, the second term was multiplied and taken into the product by the mean electricity tariff to obtain the total energy cost. The variable *psm* represents the power rating of machines in watts.

$$C_E = \left(\sum_{j=1}^{J}\sum_{s=1}^{S}\sum_{m=1}^{M} t_{jsm}\cdot p_{sm}\cdot\alpha_{jsm}\right)\times\left(\frac{Avg.\ electricity\ tariff}{60\times 1000}\right) \quad (2)$$

The maintenance cost (C_M) refers to the expenses incurred to keep machines in optimal working condition. Maintenance activities can be broadly categorized as preventive and corrective, but this study focuses exclusively on schedule-based maintenance. The maintenance cost is associated with regular servicing and upkeep based on the stipulated maximum operating time for each machine model. The operating time and corresponding cost of maintenance for each machine may vary. To determine the number of maintenance cycles required, the total operation time of machine *m* at stage *s* (*Tsm*) is divided by the maximum operating duration of machine *m* at stage *s* (*trsm*). This result is rounded up to calculate the number of maintenance cycles, as shown in equation (3).

$$C_M = \sum_{s=1}^{S}\sum_{m=1}^{M}\left\lceil\frac{T_{sm}}{tr_{sm}}\right\rceil\times r_{sm} \quad (3)$$

Delay cost (Cd) is a charge imposed on the manufacturer when they fail to deliver orders on the due date. In this work, late penalties are incurred if an industry cannot complete the requested orders on time. The late penalty is applied daily, meaning that for each day a job remains overdue, the penalty cost increases. The duration of the delay influences the magnitude of the late penalty cost. Where C_j represents the completion date of a job, and the due date is represented by D_j. The lateness factor, denoted as Y_j and defined in equation (4), quantifies the extent of the delay. The total late penalty cost, C_P, can be computed using equation (5).

$$Y_j = \left(\frac{C_j}{\text{Daily working time in minutes}}\right) - D_j$$
$$Y_j = \begin{cases} y_j & \text{if } y_j > 0 \\ 0 & \text{else} \end{cases} \quad (4)$$

$$C_d = \sum_{j=1}^{J} Y_j \times \text{delay charges per day} \quad (5)$$

3. TEACHING-LEARNING-BASED OPTIMIZATION ALGORITHM

The Teaching-Learning-Based Optimization (TLBO) algorithm is a bio-inspired metaheuristic optimization algorithm [12]. It works on the analogy of the teaching and learning process inside a classroom. This algorithm works by simulating two fundamental processes of teaching and learning: the teacher's effort to enhance the knowledge of the entire classroom by average and the learners' tries to gain knowledge from any random partner in class. The TLBO algorithm is designed to execute complex continuous optimization problems by iteratively improving a population of candidate solutions. TLBO operates in two phases: The Teacher phase and the Learner phase. The TLBO is initialized with random candidate solutions and passes through the teacher and learner phases before going to the next candidate solution. Each phase is discussed separately below based on the flowchart, as shown in Figure 1.

In the teacher phase, the teacher declares the solution corresponding to the best fitness, while all other solutions are considered by students. In essence, students learn from the teacher and try to move closer to its higher quality solution. After the Teacher phase, the Learner phase begins, where solutions randomly interact with their partners to learn from them. However, if the partner has inferior knowledge than the best student, the partner will try to learn from the best student and increase the knowledge. In this way, the algorithm generates the new candidate solutions and tries to determine the best fitness under a set of constraints. Initially, the random numbers for candidates' solutions are generated based on the formula *X= L+ rand. *(U-L).* where *L* is the lower bound, *U* is the upper bound, and *rand* represents a random number between 0 and 1.

3.1 Teacher Phase

The main aim of TLBO is to determine the updated fitness function based on the newly generated solutions for the decision variables. In the teacher phase, the expression *Xnew=Xold +r (Xbest - Tf*Xmean)* is used to calculate the new solution for the decision variables. Where *Xold* represents the current candidate solution, *r* is the random number taken between 0 and 1. The *Xbest* is known as a teacher, which corresponds to the best fitness value. *Tf* is called the teaching factor with a value of either 1 or 2, and it can be found using *Tf= round (1+rand)*. The *Xmean* is the average of decision variables for all candidates' solutions. Once *Xnew* is found within the specified range of defined bounds, the corresponding fitness value will be calculated. Subsequently, the newly computed fitness value will be compared with the existing one. If the new fitness value improves, the older fitness and decision variable will be replaced with the new one, otherwise the old values will be considered.

3.2 Learner Phase

In the Teacher phase, the updated solution will pass through the Learner phase before going to the second candidate solution. The search for the new solution *Xnew* depends on the choice of a random partner *Xp*. The comparison of fitness values is crucial in this phase, and the following expressions are employed to calculate *Xnew* based on the comparison of fitness values, as shown in equations 6 and 7.

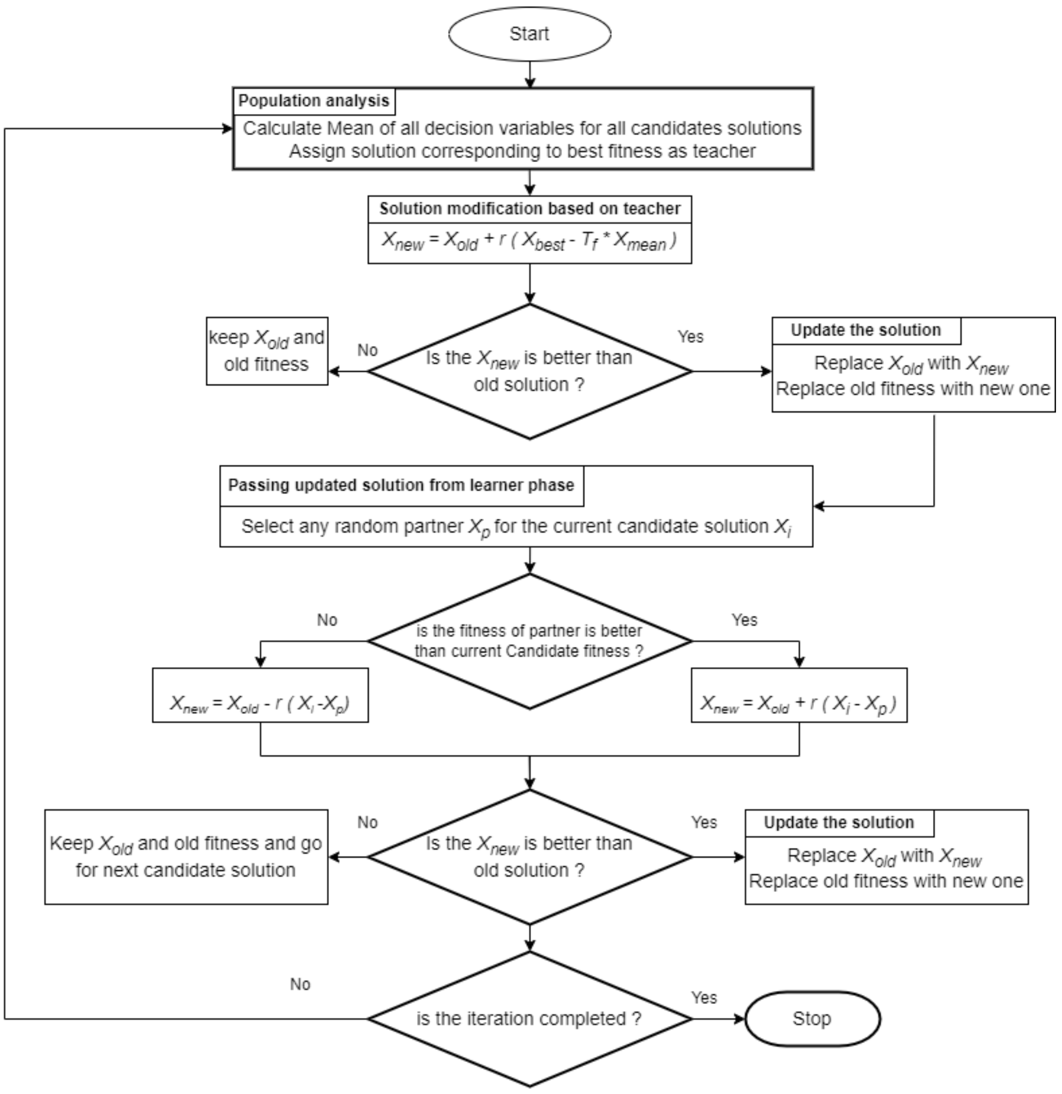

Start
Population analysis
Calculate Mean of all decision variables for all candidates solutions
Assign solution corresponding to best fitness as teacher
Solution modification based on teacher
$X_{new} = X_{old} + r\,(X_{best} - T_f * X_{mean})$
keep X_{old} and old fitness
No
Is the X_{new} is better than old solution ?
Yes
Update the solution
Replace X_{old} with X_{new}
Replace old fitness with new one
Passing updated solution from learner phase
Select any random partner X_p for the current candidate solution X_i
No
is the fitness of partner is better than current Candidate fitness ?
Yes
$X_{new} = X_{old} - r\,(X_i - X_p)$
$X_{new} = X_{old} + r\,(X_i - X_p)$
Keep X_{old} and old fitness and go for next candidate solution
No
Is the X_{new} is better than old solution ?
Yes
Update the solution
Replace X_{old} with X_{new}
Replace old fitness with new one
No
is the iteration completed ?
Yes
Stop

Figure 1. TLBO flowchart

$$If\ f > fp, then\ Xnew = Xold - r * (Xold - Xp) \tag{6}$$

$$If\ f < fp, then\ Xnew = Xold + r * (Xold - Xp) \tag{7}$$

where *Xold* represents the current candidate solution, *Xp* is the partner solution, and *r* is a random number between 0 and 1. The corresponding fitness value is determined based on *Xnew*. The greedy selection is conducted to look for new fitness value improvement. The same process will be conducted for all the candidate solutions until the termination criteria are met.

The TLBO algorithm is successfully implemented for various scheduling problems. The algorithm can explore the search space and its simple implementation makes it a suitable choice for solving these complex optimization problems. In a paper, the TLBO was tested for combinatorial optimization scheduling problems of job shops and flowshop scheduling cases [31]. Experiments showed that the TLBO has great potential when compared with the heuristic approach for scheduling problems. Besides this, another study presents a variant version of TLBO, where various weights are allocated to the "students" (candidate solutions) during the learning stage. Higher weights are allotted to the students with better solutions. Three distinct approaches were explored for assigning these weights. This TLBO was applied to shop scheduling problems [32]. The TLBO algorithm is implemented to solve the NP-hard Job Shop Scheduling Problem (JSSP) to find the optimal schedule to get the minimum makespan. TLBO's performance is appraised by solving 10 Taillard benchmark problems and comparing the results with Artificial Immune System (AIS) and Differential Evolution (DE) algorithms. The results demonstrate that TLBO is an effective evolutionary algorithm for developing the finest operation scheduling solutions for JSSP [33]. The TLBO algorithm was also used for energy optimization in smart homes, demonstrating more effective results compared to other metaheuristics [34].

Since TLBO was designed for continuous problems, cost optimization problems in scheduling are often combinatorial. Hence, a decoding scheme must be applied to TLBO to optimize CHFS scheduling problems. TLBO can be adapted for cost optimization in various other scheduling problems, such as vehicle routing, workforce scheduling, or production scheduling, by formulating the fitness function to represent the relevant cost factors specific to the problem domain. An integrated cost model was established to optimize the CHFS problems using the TLBO algorithm. While implementing TLBO, a permutation-based solution representation was encoded where each solution represents the job sequence on each machine. Then, the random candidates' solutions were generated. Consequently, the rest of the TLBO steps were applied, such as teacher, learner, and evaluation till the termination. The TLBO algorithm was chosen to solve the CHFS problem due to its proven effectiveness in tackling complex optimization problems across various domains.

4. COMPUTATIONAL EXPERIMENT SETUP

A computational experiment was performed to evaluate the performance of the TLBO algorithm in optimizing the developed (CHFS) model. The hypothetical dataset defined by Carlier and Neron was utilized for this purpose [35]. This dataset is commonly used to evaluate HFS scheduling problems. It consists of 12 test benchmark problems with varying dimensions and sizes, ranging from 10 to 15 jobs, 5 to 10 stages, and varying machine configurations across the problems. The experiment aimed to assess the effectiveness of the TLBO algorithm for the CHFS model using the diverse set of benchmark problems from the Carlier and Neron dataset. The varying problem dimensions and machine configurations provided a comprehensive test environment to evaluate the algorithm's performance and robustness. In each of the benchmark problems, the following assumptions were considered: (1) The machines available at different stages were non-identical, confirming their different power ratings; (2) The labor cost per hour was set to MYR12; (3) A penalty charge of 5 MYR per minute was imposed. These assumptions were strictly followed when feeding the benchmark problems into the CHFS model.

To check the strength and solution quality of the TLBO algorithm, comparative algorithms based on three factors were selected: popularity in the literature from 2010 to 2022, well-established metaheuristics and recent algorithms from 2021 to 2024. A total of 5 algorithms were chosen for comparison. The list of selected comparative algorithms are as follows:

- Particle Swarm Optimization (PSO) [36]
- Genetic Algorithm (GA) [37]
- Goose Algorithm (GOO) [38]
- Dwarf Mongoose Optimization Algorithm (DMOA) [39]
- Gray Wolf Optimizer (GWO) [40]

For the experiments, the population size was kept at 30 for all optimization algorithms with iterations of 300 and optimization runs of 10. The jobs' processing times were generated in the range of {3, 20}. Table 1 showcased the details of each test instance, comprising the stages, number of jobs and machines at each stage. The experiment was done using MATLAB on a Dell Latitude 7480 laptop with 8 GB RAM, a 64-bit operating system, and an Intel(R) Core (TM) i5-6300U CPU @ 2.40GHz 2.50 GHz processor.

Table 1. Benchmark test problem configurations

Test Problem	Jobs	Stages	Machines on each stage
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

Table 1. (cont.)

Test Problem	Jobs	Stages	Machines on each stage
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

5. RESULTS AND DISCUSSION

The computational experiment results for the CHFS model using benchmark problems by several proposed metaheuristics algorithms can be seen in Table 2. Based on the results obtained, none of the single algorithms dominated the performance in all indicators and problems. Some of the best results in different indicators can be observed in TLBO, GWO, and DMOA. These computational results revealed interesting insights into the performance of proposed metaheuristics. The fundamental indicators such as average fitness (AVG), standard deviation (SD), maximum fitness (Max), minimum fitness (Min) and the average computational time (CT) were determined to see the strength and efficiency of algorithms while optimizing CHFS benchmark scheduling problems. In problem " j10c5a2", j10 represents 10 jobs, c5 shows five stages, and a2 is the machine's configuration in each stage.

Table 2. Computational experimental results

Problem	Indicator	DMOA	GA	GOO	GWO	PSO	TLBO
J10c5a2	AVG	961.71	1340.38	1410.49	998.56	1199.71	975.82
	SD	110.44	208.21	266.37	132.09	235.06	71.45
	MAX	1165.66	1679.45	2048.773	1291.11	1709.72	1082.30
	MIN	768.52	964.97	1011.98	867.85	933.60	881.49
	CT	21.18	11.95	7.12	13.24	13.42	25.04
J10c5b1	AVG	911.35	1065.27	1441.15	999.49	1112.47	930.11
	SD	88.12	157.12	115.17	97.57	156.43	185.69
	MAX	1086.63	1393.21	1555.44	1136.76	1346.12	1239.55
	MIN	798.87	912.23	1211.47	861.93	834.83	631.28
	CT	20.04	12.65	8.24	12.87	11.55	27.51
J10c5c1	AVG	463.54	525.40	613.43	452.58	486.69	461.53
	SD	6.29	106.90	119.84	8.31	53.81	16.32
	MAX	477.40	792.54	904.15	466.84	587.92	504.50
	MIN	457.03	455	469.82	440.93	449.66	443.75
	CT	27.37	16.91	12.36	16.52	15.82	34.69
J10c5d1	AVG	106.85	152.04	312.83	101.44	213.59	99.30
	SD	4.23	60.85	174.27	7.67	92.43	5.52
	MAX	112.41	241.99	665.02	115.43	408.67	105.69
	MIN	97.70	104.12	104.92	88.78	98.77	91.06
	CT	39.56	17.17	11.08	16.53	15.47	36.81
J10c10a2	AVG	11202.85	10850.72	12181.50	10430.34	10942.08	9899.75
	SD	412.91	589.70	772.53	684.49	600.03	730.78
	MAX	11883.58	12154.99	13154.88	11748.67	11840.41	11269.17
	MIN	10600.40	10220.34	10846.41	9235.34	10013.02	9023.57
	CT	71.92	22.93	12.62	22.25	22.23	49.56

Table 2. (cont.)

Problem	Indicator	DMOA	GA	GOO	GWO	PSO	TLBO
J10c10b1	AVG	9681.85	9734.33	10522.60	9396.18	9428.40	8722.01
	SD	271.53	518.85	879.52	473.12	581.85	400.12
	MAX	10148.55	10720.83	11860.31	10342.97	10483.6	9219.33
	MIN	9445.84	8947.85	9020.46	8666.6	8491.04	8103.74
	CT	59.62	22.45	17.95	22.35	24.96	57.62
J10c10c1	AVG	6654.46	6255.29	6816.02	5719.11	6043.14	5864.19
	SD	291.09	387.37	536.53	738.01	581.81	810.18
	MAX	7137.11	6741.53	7595.54	6765.18	6889.40	7226.50
	MIN	6198.12	5519.83	5619.56	4579.77	5082.39	4848.28
	CT	61.22	31.16	15.12	32.54	29.76	67.18
J15c5a1	AVG	2857.31	2845.45	3118.12	2566.64	2791.69	2606.89
	SD	114.81	262.44	378.40	296.31	203.96	269.68
	MAX	3053.11	3128.07	3536.14	2996.96	3093.15	3090.35
	MIN	2701.15	2362.80	2240.55	1947.29	2372.88	2239.26
	CT	46.94	16.74	8.35	16.65	17.47	35.17
J15c5b1	AVG	5238.40	5203.49	5396.76	4650.07	4691.45	4530.68
	SD	165.67	364.05	485.63	366.39	400.98	276.19
	MAX	5457.48	5633.60	6301.83	5297.51	5521.13	4999.59
	MIN	4962.40	4594.18	4610.97	4028.24	4083.22	4181.36
	CT	29.64165	18.0835	7.89784114	4650.07	17.49	32.93
J15c5c1	AVG	2565.841	2488.744	2785.595452	2328.36	2375.23	2243.06
	SD	136.378	237.1205	247.781023	261.96	245.78	185.18
	MAX	2818.168	2713.796	3073.181108	2612.45	2874.16	2449.19
	MIN	2382.45	2064.863	2389.831925	1794.17	2092.90	1957.76
	CT	34.15145	18.52943	8.07893626	26.88	16.82	33.44
J15c10a2	AVG	10773.15	9947.449	11241.3985	8844.71	9720.67	9176.30
	SD	437.704	563.515	536.0590277	683.44	709.54	895.33
	MAX	11288.41	11007.03	11817.12483	9797.31	10660.12	10261.63
	MIN	9835.29	9054.851	10048.74753	7571.74	8277.22	7969.95
	CT	85.43099	34.48052	17.89630655	55.18	32.19	67.24
J15c10b1	AVG	11540.96	11022.27	12001.79831	10373.15	9874.39	9881.20
	SD	410.6804	678.7138	733.16275	848.84	635.51	553.54
	MAX	12117.55	11974.11	13172.03908	11689.99	10717.3	10647.55
	MIN	10595.81	9871.47	11153.96908	9191.28	8810.86	9129.43
	CT	70.03509	33.1962	17.77376022	48.11	36.65	73.44

The average fitness (AVG) is the average of all fitness after all optimization runs. This indicator represents the overall performance of an algorithm throughout all runs of optimization. Regarding average fitness (AVG), the TLBO algorithm consistently achieved best results in 42% of benchmark problems. The second-best algorithm was GWO, obtaining the best results for mean fitness in 33% of test problems, followed by DMOA in 17% of test instances. In this scenario, the algorithms with lower performance are GA, GOO, and PSO.

The second indicator is SD, which shows the spread of the obtained solution in each optimization run from average fitness. The smaller SD value of an algorithm indicates a better exploitation capability. In this category, DMOA obtained the smaller SD value in 92% of problems while TLBO in 8% of test instances. This is the reason these algorithms outperformed in maximum test problems in average fitness. In this indicator, the PSO, GOO, GWO, and GA showcased the worst solutions for all problems. The larger SD value was recorded by TLBO in problem J15c10a2. The TLBO has the best fitness in five test instances, followed by SD in one problem. The algorithms GA, GOO, PSO and GWO fail to achieve better SD.

The Max indicator is the worst solution achieved by an algorithm in all runs of optimizations. Regarding the Max indicator, TLBO obtained the best result in 58%, while GWO in 25% of test problems. The DMOA achieved better solutions for the Max indicator in only one test problem, J10c5b1. Another interesting trend is that, in some problems, the Max indicator for TLBO is even better than the min fitness of other algorithms. As an example, in problem J10c5a2, the Max indicator of TLBO is better than the min fitness of PSO. This smaller Max indicator value helps to obtain better fitness in most test problems by TLBO.

The Min indicator is the best solution among all obtained fitness in all optimization runs, which is discovered by an algorithm. This indicator shows the effectiveness of an algorithm while optimizing a problem. GWO obtained the best solution for the Min indicator in 7 problems out of 12, followed by TLBO in 3 problems. The GWO failed to achieve the Min value in problems J10c5b1, J10c5c1, and J15c10b1, respectively. In the Min indicator, TLBO in problem J10c5b1 achieved the best solution. The algorithms PSO, GA, and GOO remained ineffective regarding the Min indicator.

The final indicator is CT, which is the average time taken by an algorithm to complete one optimization run. The GOOSE algorithm (GOO) obtained a smaller value of CT in all test problems. This shows the high convergence and low exploration capability of the GOO algorithm. It was noticed that TLBO and DMOA recorded the larger CT in all test problems, which clarifies the better exploration capability while searching for better and quality solutions. Besides this, GWO, PSO, and GA also showed relatively smaller CT as in some benchmark problems.

The performance ranking based on the average fitness of TLBO can be seen in Table 3. It was observed that TLBO secured the first position in the average performance ranking, followed by GWO. The PSO, DMOA, GA and ACO obtained the third, fourth, fifth and sixth ranking positions in terms of average fitness ranking, respectively.

Table 3. Performance ranking based on the average fitness of TLBO

Problem	DMOA	GA	GOO	GWO	PSO	TLBO
J10c5a2	1	5	6	3	4	2
J10c5b1	1	4	6	3	5	2
J10c5c1	3	5	6	1	4	2
J10c5d1	3	4	6	2	5	1
J10c10a2	5	3	6	2	4	1
J10c10b1	4	5	6	2	3	1
J10c10c1	5	4	6	1	3	2
J15c5a1	5	4	6	1	3	2
J15c5b1	5	4	6	2	3	1
J15c5c1	5	4	6	2	3	1
J15c10a2	5	4	6	1	3	2
J15c10b1	5	4	6	3	1	2
Avg Rank	3.9	4.1	6	1.9	3.4	1.5

The head-to-head comparison for TLBO performance can also be seen in below Table 4. The table revealed that TLBO obtained the worse solutions in two problems when compared with DMOA, while in one problem with PSO. In the rest of the test problems, none of the algorithms obtained a better solution than TLBO. In short, it was concluded that, in 88.3% of problems, TLBO performed well while providing the worst solutions in 11.7% of test instances.

Table 4. Head-to-head comparison for TLBO performance

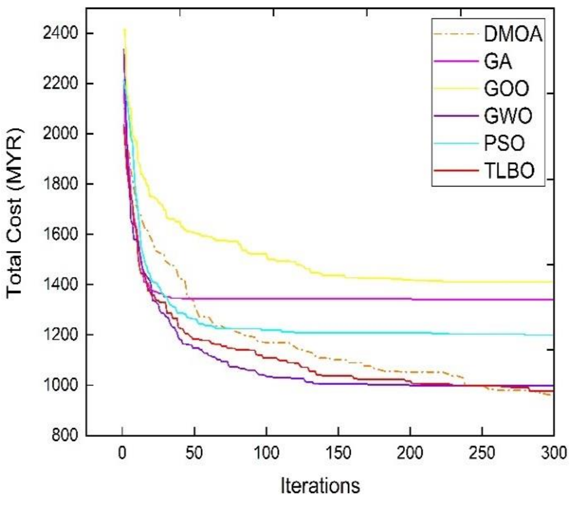
TLBO Performance	DMOA	GA	GOO	GWO	PSO	(%)
Better	10	12	12	8	11	88.3%
Equal	0	0	0	0	0	0%
Worse	2	0	0	4	1	11.7%

The Wilcoxon rank-sum test was utilized to assess the significance of the results obtained by TLBO. Its purpose was to ascertain whether a notable distinction existed between the fitness values derived from TLBO and those of compared algorithms. The null hypothesis posits that no significant distinction exists between the two sets, while the alternative hypothesis suggests otherwise. If the resulting p-value surpasses the predetermined significance level (alpha, typically set at 0.05), the null hypothesis will be delegated. Conversely, if the p-value falls below alpha, the null hypothesis will be rejected.

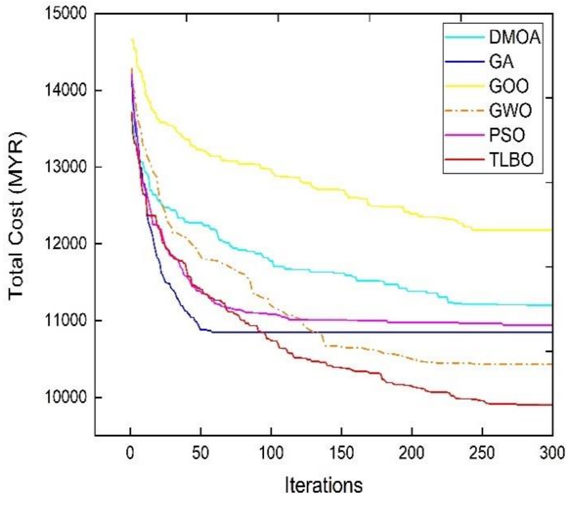
The p-value was generated by the Wilcoxon rank-sum test, as shown in Table 5. It was observed that 58% of the p-values were found below 0.05. This means that in 58% of the problems, there exists a substantial difference between TLBO fitness and other proposed algorithms. The algorithms that struggled to achieve almost the same fitness as TLBO in some problems are denoted by (‘*’). In 41% of the problems, the p-value was found above alpha (0.05), meaning that the fitness of these comparative algorithms was almost aligned with TLBO. Almost all the algorithms contributed to produce deemed solutions in 1 to 2 test instances, although GWO and PSO have greater contributions regarding these insignificant results in 6 benchmark problems.

Table 5. Wilcoxon rank-sum test (p-value)

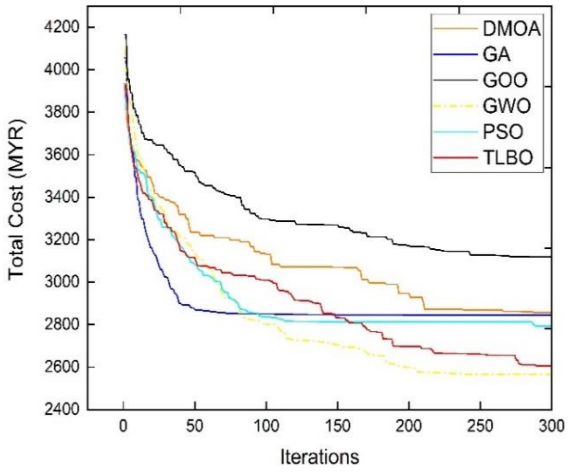
Problem	DMOA	GA	GOO	GWO	PSO
J10c5a2	0.8501*	7.6854e-04	4.3964e-04	1*	0.0073
J10c5b1	0.8501*	0.0757*	2.4613e-04	0.3075*	0.0452
J10c5c1	0.1859*	0.0452*	3.2984e-04	0.0757*	0.1859*
J10c5d1	0.0036	3.2984e-04	2.4613e-04	0.6776*	0.0036
J10c10a2	0.0022	0.0073	5.8284e-04	0.1405*	0.0091
J10c10b1	1.8267e-04	7.6854e-04	3.2984e-04	0.0036	0.0073
J10c10c1	0.0211	0.3447*	0.0058	0.7337*	0.4727*
J15c5a1	0.0211	0.1212*	0.0058	0.8501*	0.0640*
J15c5b1	2.4613e-04	0.0013	7.6854e-04	0.4274*	0.1859*
J15c5c1	5.8284e-04	0.0312	7.6854e-04	0.3847*	0.2730*
J15c10a2	5.8284e-04	0.0452	3.2984e-04	0.3847*	0.1620*
J15c10b1	2.4613e-04	1.8267e-04	1.8267e-04	0.1212*	0.9097*



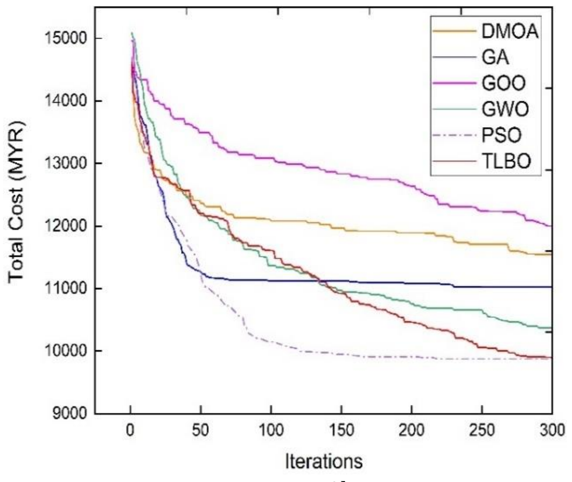
J10c5a2



J10c10a2



J15c5a1



J15c10b1

Figure 2. Convergence profiles for CHFS optimization

The convergence profiles for various sizes and dimensions of test problems have been showcased in Figure 2. The convergence curves reveal some of the distinct features of various proposed metaheuristics algorithms that have been applied to CHFS problems. In this work, six algorithms were implemented on four variant problems, and their convergences have been plotted against each other. The TLBO algorithm keeps consistent convergence in all four test problems till the end of iteration over other algorithms. It was noted that the GWO struggled to achieve the same results as TLBO. Such behaviors of this algorithm declare stable and reliable optimization outcomes. The GA fitness falls down suddenly before 50 iterations and then passes through the optimal trap region. The rapid fall before 100 iterations and then passing through a steady state path declares the stable refinement and exploration capability of these algorithms to find the better solution in search space. Such algorithms include DMOA, PSO and GWO. These convergence profiles aid the researchers, who optimize various optimization problems to choose the most stable and efficient algorithm. To obtain an appealing structure for convergence plots based on various algorithms, keep the number of iterations from 300 to 500 and optimization runs of 10 to 20.

There are some theoretical reasons behind TLBO's better performance. The TLBO algorithm's superior performance stems from its efficient balance of exploration and exploitation, achieved through its Teacher and Learner phases. In the Teacher Phase, the best solution influences the entire population, enhancing convergence speed, while the Learner Phase promotes diversity by enabling individuals to learn from each other, preventing premature convergence. TLBO's parameter-free nature simplifies implementation and use, reducing the need for fine-tuning, which is common in other algorithms. Its robustness and adaptability to various optimization problems, combined with favorable convergence characteristics and a synergistic blend of global and local search capabilities, further enhance its effectiveness. As a population-based approach, TLBO leverages multiple solutions concurrently, aiding in avoiding local optima and ensuring a thorough global search, making it highly effective for complex optimization problems.

5.0 CONCLUSIONS

This study presents the cost-based hybrid flowshop scheduling (CHFS) optimization problem using the Teaching Learning-Based Optimization (TLBO) algorithm. The CHFS problem model is established for the objective function of minimizing total production costs such as labor, energy, maintenance, and delay. The TLBO algorithm was implemented on CHFS scheduling problems. The TLBO algorithm was selected due to its inherent ability to strike a favorable balance between exploration and exploitation, enabling it to efficiently identify near-optimal scheduling solutions. The proposed TLBO algorithm was utilized, and the performance was investigated through computational experiments. The results demonstrated that TLBO was able to outperform other popular metaheuristic algorithms like Grey Wolf Optimizer, Genetic Algorithm, and Particle Swarm Optimization in terms of solution quality and convergence speed. The TLBO achieved the best results in 42% of the cases, while the remaining 58% achieved the second-best rank, which demonstrates the consistency of its performance. The study showcases the potency of the TLBO algorithm in optimizing intricate CHFS scheduling problems. The experimental findings underscore the potential of the TLBO algorithm to assist with production scheduling and planning in manufacturing systems.

There are certain limitations that suggest opportunities for future work. Future research could explore multiobjective formulations, and TLBO implementation in real-world case studies could further validate its practical applicability. Investigating the hybridization of TLBO with other optimization techniques or the incorporation of problem-specific heuristics could potentially enhance the algorithm's performance for larger and more complex CHFS problems. The findings from this research can empower manufacturing managers and production planners to make more informed, data-driven decisions regarding their scheduling strategies.

ACKNOWLEDGEMENTS

The authors are thankful to Universiti Malaysia Pahang Al-Sultan Abdullah for funding this research under RDU223017 research grant**.**

CONFLICT OF INTEREST

The authors declared that they do not have any conflicts of interest.

AUTHORS CONTRIBUTION

W. Ullah: Numerical experiment, result analysis, draft writing.

M.A.N Mu’tasim: Coding, manuscript improvement.

M.F.F. Rashid: Problem modeling, manuscript improvement.

REFERENCES

[1] S. Khuri, T. Bäck, and J. Heitkötter, “An evolutionary approach to combinatorial optimization problems,” Proceedings of the 22nd annual ACM Computer Science Conference on Scaling up: meeting the challenge of complexity in real-world computing applications meeting the challenge of complexity in real-world computing applications - CSC ’94, 1994.

[2] D. Istokovic, M. Perinic, M. Vlatkovic, and M. Brezocnik, “Minimizing total production cost in a hybrid flow shop: A simulation-optimization approach,” *International Journal of Simulation Modelling*, vol. 19, no. 4, pp. 559–570, 2020.

[3] Z. Han, X. Ma, L. Yao, and H. Shi, “Cost optimization problem of hybrid flow-shop based on PSO algorithm,” in *Advanced Materials Research*, 2012, pp. 1616–1620, 2012.

[4] E. Osaba, E. Villar-Rodriguez, J. Del Ser et al., “A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems,” *Swarm and Evolutionary Computation*, vol. 64, p. 100888, 2021.

[5] J.N.D. Gupta, K. Krüger, V. Lauff, F. Werner, and Y.N. Sotskov, "Heuristics for hybrid flow shops with controllable processing times and assignable due dates," *Computers & Operations Research*, vol. 29, no. 10, pp. 1417–1439, 2002.

[6] A. Janiak, E. Kozan, M. Lichtenstein, and C. Oğuz, "Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion," *International Journal of Production Economics*, vol. 105, no. 2, pp. 407–424, 2007.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[8] Y. Wang, J. Tang, Z. Pan, and C. Yan, "Particle swarm optimization-based planning and scheduling for a laminar-flow operating room with downstream resources," *Soft Computing*, vol. 19, no. 10, pp. 2913–2926, 2015.

[9] S. Schulz, "A genetic algorithm to solve the hybrid flow shop scheduling problem with subcontracting options and energy cost consideration," in *Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology–ISAT 2018: Part III*, pp. 263-273. Springer International Publishing, 2019.

[10] Z. Han, X. Meng, B. Ma, and C. Wang, "Cost optimization problem of hybrid flow-shop based on differential evolution algorithm," in *Advanced Materials Research*, 2012, pp. 1692–1700, 2012.

[11] E.A. Khan and M.K. Shambour, "An optimized solution for the transportation scheduling of pilgrims in Hajj using harmony search algorithm," *Journal of Engineering Research*, vol. 11, no. 2, p. 100038, 2023.

[12] R.V. Rao, V.J. Savsani, and D.P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *CAD Computer Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[13] L. Fanjul-Peyro, F. Perea, and R. Ruiz, "Models and metaheuristics for the unrelated parallel machine scheduling problem with additional resources," *European Journal of Operational Research*, vol. 260, no. 2, pp. 482–493, 2017.

[14] Y. Fei and H. Ma, "Multiobjective joint optimization of batch-discrete hybrid flow shop scheduling integrated with machine maintenance," In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pp. 247-253, 2018.

[15] W. Songserm and T. Wuttipornpun, "MIP-based heuristic algorithm for finite capacity MRP problem in hybrid flow shop with unrelated parallel machines," International Journal of Industrial and Systems Engineering, vol. 33, no. 2, p. 181, 2019.

[16] X. Lian, Z. Zheng, C. Wang, and X. Gao, "An energy-efficient hybrid flow shop scheduling problem in steelmaking plants," *Computers & Industrial Engineering*, vol. 162, p. 107683, 2021.

[17] W. Shao, Z. Shao, and D. Pi, "A network memetic algorithm for energy and labor-aware distributed heterogeneous hybrid flow shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 75, p. 101190, 2022.

[18] W. Sukkerd, J. Latthawanichphan, T. Wuttipornpun, and W. Songserm, "Non-population metaheuristics with a new fitness evaluation to minimise total penalty costs for a hybrid flow shop with assembly operations scheduling problem," in *Proceedings - 2021 Research, Invention, and Innovation Congress: Innovation Electricals and Electronics, RI2C 2021*, Institute of Electrical and Electronics Engineers Inc., Sep. 2021, pp. 340–346, 2021.

[19] X. An, G. Si, T. Xia, D. Wang, E. Pan, and L. Xi, "An energy-efficient collaborative strategy of maintenance planning and production scheduling for serial-parallel systems under time-of-use tariffs*," Applied Energy*, vol. 336, p. 120794, 2023.

[20] A. Ziaeifar, R. Tavakkoli-Moghaddam, and K. Pichka, "Solving a new mathematical model for a hybrid flow shop scheduling problem with a processor assignment by a genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 1–4, pp. 339–349, 2011.

[21] X.K. Tian, B.C. Ma, J.W. Zhang, R.Y. Zhao, and J. Wang, "Research on multi-objective optimization and simulation to HFSSP hybrid flow-shop scheduling problem for energy saving," *IOP Conference Series: Materials Science and Engineering*, vol. 504, p. 012108, 2019.

[22] P. Li, Q. Xue, Z. Zhang, J. Chen, and D. Zhou, "Multi-objective energy-efficient hybrid flow shop scheduling using Q-learning and GVNS driven NSGA-II," *Computers & Operations Research*, vol. 159, p. 106360, 2023.

[23] M. Tsiftsoglou, M. Marinaki, and I. Marinakis, "Addressing the single and multiobjective energy-aware flowshop scheduling problem through diverse variations of the PSO algorithm," Available at SSRN 4798449, 2024.

[24] K. Worasan, K. Sethanan, R. Pitakaso, T. Jamrus, K. Moonsri, and P. Golinska-Dawson, "A hybridization of PSO and VNS to solve the machinery allocation and scheduling problem under a machinery sharing arrangement," *Intelligent Systems with Applications*, vol. 18, p. 200206, 2023.

[25] J. Du, J. Mumtaz, and J. Zhong, "Improved spider monkey optimization algorithm for hybrid flow shop scheduling problem with lot streaming," *Engineering Proceedings*, vol. 45, no. 1, p. 23, 2023.

[26] Z. Han, X. Dong, and S. Lin, "An effective DE algorithm for hybrid flow shop load balancing scheduling problem," *Advances in Computer Science Research*, pp. 213-217, Atlantis Press, 2015.

[27] C. Song, "A self-adaptive multiobjective differential evolution algorithm for the unrelated parallel batch processing machine scheduling problem," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 5056356, 2022.

[28] D. Lei, J. Zhang, and H. Liu, "An adaptive two-class teaching-learning-based optimization for energy-efficient hybrid flow shop scheduling problems with additional resources," *Symmetry*, vol. 16, no. 2, p. 203, 2024.

[29] J.N. Shen, L. Wang, and H.Y. Zheng, "A modified teaching-learning-based optimisation algorithm for bi-objective re-entrant hybrid flowshop scheduling," *International Journal of Production Research*, vol. 54, no. 12, pp. 3622–3639, 2016.

[30] D. Lei, L. Gao, and Y. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Transactions on Engineering Management*, vol. 65, no. 2, pp. 330–340, 2018.

[31] A. Baykasoğlu, A. Hamzadayi, and S. Y. Köse, "Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases," *Information Sciences*, vol. 276, pp. 204–218, 2014.

[32] L.R. Rodrigues and J.P. Pordeus Gomes, "TLBO with variable weights applied to shop scheduling problems," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 3, pp. 148–158, 2019.

[33] K.N. Reddy and G. Padmanabhan, "Teaching learning based optimization (TLBO) for job shop scheduling problems," In *First International Conference on Productivity, Efficiency and Competitiveness in Design and Manufacturing, Coimbatore, Tamilnadu, India*, pp. 444 – 449, 2016.

[34] A. Manzoor et al., "User comfort oriented residential power scheduling in smart homes," In *Innovative Mobile and Internet Services in Ubiquitous Computing: Proceedings of the 11th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2017)*, pp. 171-180. Springer International Publishing, 2018.

[35] J. Carlier and E. Neron, "An exact method for solving the multi-processor flow-shop," *RAIRO-Operations Research-Recherche Opérationnelle*, vol. 34, no. 1, pp. 1–25, 2000.

[36] F. Marini and B. Walczak, "Particle swarm optimization (PSO): A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015.

[37] T.V Mathew, "Genetic Algorithm," *Report submitted at IIT Bombay*, vol. 53, 2012.

[38] R.K. Hamad and T.A. Rashid, "GOOSE Algorithm: A powerful optimization tool for real-world engineering challenges and beyond," *Evolving Systems*, vol. 15, pp. 1249–1274, 2024.

[39] J.O. Agushaka, A.E. Ezugwu, and L. Abualigah, "Dwarf mongoose optimization algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 391, p. 114570, 2022.

[40] S. Mirjalili, S.M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.