Soft Computing (2024) 28:1459–1489
https://doi.org/10.1007/s00500-023-08342-2

APPLICATION OF SOFT COMPUTING

An optimization algorithm for the multi-objective flexible fuzzy job shop environment with partial flexibility based on adaptive teaching–learning considering fuzzy processing times

Mary Jiménez Tovar[1] · Jaime Acevedo-Chedid[1] · Holman Ospina-Mateus[1] · Katherinne Salas-Navarro[2] · Shib Sankar Sana[3]

Accepted: 27 April 2023 / Published online: 5 June 2023

Abstract
Production scheduling is a critical factor to enhancing productivity in manufacturing engineering and combinatorial optimization research. The complexity and dynamic nature of production systems necessitates innovative solutions. The Job Shop Flexible Programming Problem (FJSP) provides a realistic environment for production, where processing times are variable and uncertain, and multiple objectives need optimization. To solve the Multi-Objective Flexible Fuzzy Job Shop problem with partial flexibility (P-MOFfJSP), this paper proposes a hybrid metaheuristic approach that combines the Teaching–Learning-based Optimization (TLBO) algorithm with a Genetic Algorithm. The proposed algorithm of Adaptive TLBO (TLBO-A) uses two genetic operators (mutation and crossover) with an adaptive population reconfiguration strategy, ensuring solution space exploration and preventing premature convergence. We have evaluated the TLBO-A algorithm's performance on benchmark instances commonly used in programming problems with fuzzy variables. The experimental analysis indicates significant results, demonstrating that the adaptive strategy improves the search for suitable solutions. The proposed algorithm (TLBO-A) exhibits low variations (around 11%) compared to the best mono-objective heuristic for the fuzzy makespan problem, indicating its robustness. Moreover, compared with other heuristics like traditional TLBO, the variations decrease to around 1%. However, TLBO-A stands out as it aims to solve a multi-objective problem, improving the fuzzy makespan, and identifying good results on the Pareto frontier for the fuzzy average flow time, all within this low variation margin. Our contribution addresses the challenges of production scheduling in fuzzy time environments and proposes a practical hybrid metaheuristic approach. The TLBO-A algorithm shows promising results in solving the P-MOFfJSP, highlighting the potential of our proposed methodology for solving real-world production scheduling problems.

Keywords Flexible job-shop · Teaching–learning-based optimization · Fuzzy processing times · Hybrid algorithm

1 Introduction

The production scheduling problem is a common challenging issue faced by all production systems. It is an optimization problem that involves decisions making about machine allocation and job sequencing (Pinedo 2005) problems. Adequate production scheduling aims to account for multiple constraints to achieve timely and efficient production (Lei et al. 2017). Two significant optimization challenges in manufacturing systems are the job shop problem and the flexible problem (Zhang et al. 2022). The job shop problem involves jobs scheduling to be processed on a collection of machines, each requiring a unique sequence of machine operations (Brandimarte 1993). On the other hand, the flexible problem considers jobs assignment to machines in a way that maximizes the utilization of resources and reduces the overall production time (Song and Liu 2022).

Flexible production systems can address various complexities identifying job processing times accurately (Acevedo-Chedid et al. 2021). In this direction, analyzing uncertainty in processing times in a flexible environment is essential for optimizing production schedules in production systems (Acevedo Chedid et al. 2020). Optimal scheduling

Extended author information available on the last page of the article

can determine the success or failure of an organization, impacting its profitability and quality of services (Kundakcı and Kulak 2016). The flexible job-shop scheduling problem may involve various constraints like machine capacity, time lags, holding times, and setup times (Boyer et al. 2021). These problems have been widely studied in the academic literature and have numerous practical applications in the manufacturing industry like assembly line production, manufacturing cell scheduling, and automated warehouse management (Boyer et al. 2021).

Although traditional scheduling problems typically assume that the processing time of each operation is a fixed value in real-world manufacturing systems, the processing time can be challenging to determine precisely in advance (Fazel Zarandi et al. 2020). Additionally, completion times may be obtained ambiguously, leading to uncertainty in the scheduling process. As a result, there is a clear need to incorporate the fuzzy concept into scheduling problems (Abdullah and Abdolrazzagh-Nezhad 2014). By considering the fuzziness in both processing time and completion time, manufacturers can create more robust scheduling algorithms that better account for the complexities of real-world production systems (Seyyedi et al. 2021). For example, considering fuzzy aspects in combination with hybrid algorithms is one of the new challenges in the literature (Kisi et al. 2023).

This study aims to explore innovative solutions to tackle the challenges of production scheduling in flexible production systems with potential implications for improving the efficiency and effectiveness of production processes. The proposed solution approach is an Adaptive Teaching–Learning-based Optimization Algorithm (TLBO-A) for solving the Flexible Job Shop Problem with partial flexibility, focusing on minimizing fuzzy makespan and fuzzy average flow. This research aims to develop a metaheuristic approach combining adaptive teaching–learning strategies with optimization techniques.

The practical relevance of modeling flexible manufacturing systems and the problem of production scheduling is a topic of great interest to researchers, and more and more manufacturing industries are investing in improving their production scheduling models. The identified advantages make it clear that as long as the organization's strategy can be visualized and executed effectively in production programming, customer loyalty can be achieved, costs can be reduced, and high-quality products can be guaranteed.

This research presents a multi-objective model that considers the uncertainty of system processes through fuzzy processing times. As with any Flexible Job Shop problem, the routing flexibility associated with the jobs and heterogeneous characteristics of the machines available in each job center is considered. The optimization objectives are expressed in a linear sum of weights, and the objectives to be evaluated are: Minimize the maximum fuzzy completion time of all the jobs processed in the system $(\text{Min}\,\text{Cmax})$ and minimize the average flow time. This approach guarantees that the assignments of the jobs to the machines are carried out in a balanced way, simultaneously minimizing the maximum permanence time of the jobs in the system.

The rest of the paper is organized as follows: Sect. 2 of this paper provides a comprehensive literature review of existing solution methods, followed by a detailed description of the study problem in Sect. 3. Section 4 elaborates on the design of the proposed metaheuristic algorithm, and in Sect. 5, the obtained results and managerial implications are presented. Finally, Sect. 6 will conclude the study and discuss potential future works in this field of research.

2 Literature review

The Flexible Job Shop Scheduling Problem (FJSP) is an extension of the classic Job-Shop Scheduling Problem (JSP), which is commonly encountered in companies and is known to be an NP-hard problem (Gao et al. 2007; Kaplanoğlu 2016). The FJSP involves the flexibility of job routes among the available machines, requiring adequate sequencing, assignment, and routing of jobs (Chiang and Lin 2013). The flexibility in FJSP can be categorized as partial or total. In the case where all operations can be processed on all machines in the system, it is referred to as Total Flexible Job Shop Scheduling Problem (T-FJSP); otherwise it would be considered as a Partial Flexible Job Shop Scheduling Problem (P-FJSP) (Li et al. 2010).

Various solution methodologies have been proposed for FJSP, including conventional programming methods like exact and heuristic approaches, but their effectiveness are limited due to the combinatorial complexity associated with the problem (Chen et al. 2012; Jia and Hu 2014). The FJSP solution methods can be classified into two approaches: integration and hierarchy. The integration approach involves simultaneous sequencing and assignment, while the hierarchical approach decomposes the problem. Although the hierarchical approach may be easier to solve, the integrative approach often yields better results (Pezzella et al. 2008).

Brandimarte (1993) article pioneered a hierarchical approach to solving the FJSP. Then, various studies have been conducted to address the challenges of FJSP. Mastrolilli and Gambardella (2000) developed a taboo search heuristic with two neighborhood search functions, while

Kacem et al. (2002) focused on resource allocation and used an evolutionary control approach to minimize the makespan, the flow of the most loaded machine, and the total flow of machines. Xia and Wu (2005) proposed a hybrid algorithm that combined PSO and simulated annealing as a local search algorithm to minimize makespan. Gao et al. (2007) developed a hybrid genetic algorithm with a local bottleneck search, and Gao et al. (2008) suggested a hybrid genetic algorithm with a variable neighborhood search. Xing et al. (2009) proposed a local search algorithm that combined different dispatch rules for the multi-objective FJSP with partial and complete flexibility. Finally, Zhang et al. (2011) investigated a tabu search and particle swarm optimization algorithm to address the multi-objective FJSP.

The FJSP has been tackled by several researchers using various metaheuristic algorithms. Wang et al. (2012a, b) studied an improved Pareto-based artificial bee colony algorithm, while Chiang and Lin (2013) proposed a multi-objective evolutionary algorithm that employed the Pareto approach to minimize the makespan, total flow, and maximum flow. Baykasoğlu et al. (2014) presented a solution approach for the FJSP based on a linguistically-based metaheuristic model. Li and Gao (2016) employed a hybrid algorithm that combined guided taboo search and genetic algorithm techniques. Similarly, Xu et al. (2016) presented a memetic algorithm that minimized makespan and total flow. Deng et al. (2017) developed a hybrid evolutionary bee algorithm with a genetic algorithm that minimized makespan and total and maximum flow. Gaham et al. (2018) studied an effective harmonic search which was permutation-based. Ertenlice and Kalayci (2018) executed a Kalman multi-objective heuristic algorithm that considered makespan, total flow, and critical workflow. More recently, Braune et al. (2022) developed a genetic programming approach that considered job sequencing and machine assignment in the FJSP.

Furthermore, some authors addressed uncertainty in the FJSP considering random jobs or machine breakdowns. Zhang et al. (2022) studied the dynamic flexible job shop problem (DFJSP) and proposed a two-stage algorithm that reduced makespan and the frequencies of machine breakdowns. Lei et al. (2022) included a Markov Decision Process approach in Multi-Proximal Policy Optimization Algorithm (multi-PPO). Basiri et al. (2020) considered a fuzzy processing times multi-objective model for flexible job shop scheduling and developed a metaheuristic that combines a genetic algorithm with SAW/TOPSIS methods.

Generally speaking, the completion of real systems, processing, and delivery times are often uncertain and subject to variation is essential. To address these issues, the Fuzzy Job Shop Programming Problem (fJSP) extends the JSP by introducing fuzzy variables, such as, processing times, delivery times, and constraints (Abdullah and Abdolrazzagh-Nezhad 2014). The fJSP can be divided into three main categories: the fJSP with fuzzy delivery times, the fJSP with fuzzy processing times, and the fJSP with fuzzy processing times and delivery times. Moreover, the fuzzy Job Shop problem has also been explored in flexible environments, where additional optimization criteria are considered.

The Job Shop Flexible programming problem with fuzzy behavior variables has become an increasingly popular option for tackling scheduling problems. Researchers have shown a growing interest in this problem in recent years and have proposed various solutions, albeit still in the early stages of study (Liu et al. 2015). Different models have been proposed from the classification enunciated for the fJSP, which is considered NP-Hard. The contributions of several authors in the fJSP research works are summarized in Table 1 as follows.

The current research falls under the programming environments of production in fuzzy settings for flexible job-shop scenarios. The structure of the study is detailed below. The fFJSP assumes that all jobs are initially independent and available for processing. Likewise, a machine can process only one operation, and a job can be processed by only one machine. Also, setup times are included in the processing times, and once an operation begins, it cannot be interrupted.

In this study, an Adaptive Teaching–Learning-based Optimization Algorithm (TLBO-A) is proposed to address the P-MOfFJSP with partial flexibility, where the objective is to minimize both the fuzzy makespan and the fuzzy average flow. The fuzzy behavior is modeled using Triangular Fuzzy Numbers (TFN). The practical relevance of modeling flexible manufacturing systems and production scheduling problems has caught the attention of the researchers. Many manufacturing industries are improving their production scheduling models to reduce costs and guarantee high-quality products and thus are achieving customer loyalty. Production scheduling models have been applied in various industries, such as power generation, food processing, chemical processing, automotive factories, textile industries, aeronautical industries, and steel processing (Behnamian 2017).

This study builds upon the models developed by Zhang et al. (2011), Wang et al. (2017), Kaplanoğlu (2016), Deng et al. (2017) and Kato et al. (2018), who explored multi-objective models for the Job Shop flexible scheduling problem with fuzzy processing times. The proposed models consider the routing flexibility associated with the jobs and

Table 1 Studies related to fJSP

Author	Applied technique	Fuzzy conditions	Type objective
Wang et al. (2012a, b)	Genetic algorithm with efficient crossover and mutation operators	Dynamic processing times	Multi-objective
Zheng and Li (2012)	Artificial bee colony with neighborhood structure based	Fuzzy processing times	Mono-objective (makespan)
Wang et al. (2012a, b)	An effective bi-population-based estimation of distribution algorithm (BEDA)	Fuzzy processing times	Mono-objective (makespan)
Li et al. (2012)	Particle swarm optimization with tabu search	Fuzzy processing times	Multi-objective (completion time and makespan)
Li and Pan (2013a, b)	Hybrid chemical-reaction optimization (HCRO) algorithm with Tabu search	Fuzzy processing time (maintenance activities)	Mono-objective (makespan)
Li and Pan (2013a, b)	Particle swarm optimization and Tabu search	Fuzzy processing time	Multi-objective (completion time and makespan)
Wang et al. (2013)	Non–dominated sorting Genetic algorithm	Dynamic processing times	Multi-objective (completion time, average agreement index and minimal agreement index)
Engin et al. (2013)	A Scatter search and hybrid genetic algorithm	Uncertain processing times and due dates	Mono-objective (value of the Agreement Index)
He et al. (2013)	Swarm optimization with genetic algorithm	Processing time and deadline	Mono-objective (makespan)
Palacios et al. (2013)	Hybrid algorithm with a memetic algorithm	Uncertainty in task durations	Mono-objective (makespan)
Tran et al. (2014)	Multi-objective genetic algorithm	Processing times with uncertainty	Multi-objective (optimizes durations)
Gao et al. (2015)	A discrete harmony search (DHS) algorithm	Fuzzy processing time	Mono-objective (fuzzy completion time)
Thammano and Teekeng (2015)	Metaheuristic hybrid algorithm with the Tabu list	Fuzzy processing time	Mono-objective (fuzzy makespan)
Liu et al. (2015)	Fast estimation of distribution algorithm	Fuzzy processing time	Mono-objective (makespan)
Xu et al. (2015)	Effective Teaching–learning-based optimization algorithm	Fuzzy processing time	Mono-objective (makespan)
Palacios et al. (2015)	Co-Evolutionary Cooperative Algorithm	Fuzzy processing time	Mono-objective (fuzzy termination time)
Gao et al. (2016a, b)	An improved artificial bee colony (IABC) algorithm	Fuzzy processing time	Multi-objective (fuzzy completion time and fuzzy machine workload)
Gao et al. (2016a, b)	A two-stage artificial bee colony (TABC) algorithm	Fuzzy processing time and new job	Mono-objective (fuzzy completion time)
Wang et al. (2016)	A hybrid algorithm HICATS combines a discrete imperialist competition algorithm and Tabu search	Fuzzy processing time and fuzzy due date	Multi-objective (agreement index—fuzzy due date and completion time)
Wang et al. (2017)	Multi-objective Memetic Algorithm	Fuzzy processing times and delivery times	Mono-objective (fuzzy makespan and mean compliance rate)
Lin (2019)	Backtracking search-based hyper-heuristic	Fuzzy processing time	Mono-objective (makespan)
Lin et al. (2019)	Multi-universe optimization algorithm	Processing time and deadline	Mono-objective (fuzzy makespan)
Sun et al. (2019)	Hybrid cooperative coevolution algorithm (hCEA) based on particle swarm and genetic algorithm optimization	Fuzzy processing times	Mono-objective (minimize the fuzzy makespan)
Gao et al. (2020)	Classic differential evolution algorithm	Processing time and deadline	Mono-objective (makespan)

Table 1 (continued)

Author	Applied technique	Fuzzy conditions	Type objective
Shi et al. (2020)	Immune genetic algorithm	Fuzzy delivery time with machine failure	Multi-objective (energy, makespan, satisfaction)
Basiri et al. (2020)	A hybrid intelligent genetic algorithm	Fuzzy processing time	Mono-objective (fuzzy makespan)
Li et al. (2021)	The artificial immune system (IAIS) algorithm	A type-2 fuzzy logic system	Mono-objective (fuzzy completion time)
Seyyedi et al. (2021)	A Non-dominated Sorting Genetic Algorithm II (NSGA II)	Fuzzy processing time	Mono-objective (makespan, machine workload, and earliness /tardiness penalty)
Ortíz-Barrios et al. (2021)	Dispatching algorithm based on fuzzy AHP (FAHP) and TOPSIS	Criteria weights under uncertainty	Throughput products for earlier delivery
Wang et al. (2022)	A hybrid adaptive differential evolution (HADE) algorithm	Fuzzy processing time and completion time	Multi-objective (completion time, delay time, and energy consumption)
Lei et al. (2022)	Multi-Proximal Policy Optimization Algorithm on Markov Decision Process	Fuzzy processing time	Mono-objective (makespan)
Zhang et al. (2022)	A two-stage algorithm based on a neural network	Fuzzy processing time	Mono-objective (makespan)
Pan et al. (2022)	a bi-population evolutionary algorithm with feedback (FBEA)	Uncertainty into energy-efficient	Multi-objective (fuzzy makespan, energy consumption and minimum agreement index)
Seck-Tuoh-Mora et al. (2022)	A global neighborhood with a hill-climbing algorithm (GN-HC)	Fuzzy processing times	Mono-objective (makespan)
Li et al. (2022a, b, c)	A two-stage knowledge-driven evolutionary algorithm (TS-KEA)	Fuzzy processing times	Mono-objective (makespan)
Li et al. (2022a, b, c)	A reinforcement learning (RL)—evolutionary algorithm based on decomposition MOEA/D	Fuzzy processing time	Multi-objective (makespan and machine workload)
Song and Liu (2022)	Quantum cat swarm optimization (QCSO) algorithm	Processing time and deadline	Mono-objective (makespan)
Li et al. (2022a, b, c)	A hybrid self-adaptive multi-objective evolutionary algorithm based on decomposition (HPEA)	Fuzzy processing times	Multi-objective (makespan and workload)

account for heterogeneous machine characteristics at each work center. The optimization objectives are expressed as a linear sum of weights, aiming to minimize the maximum fuzzy completion time of all processed jobs (Min Cmax) and the average flow time, ensuring a balanced assignment of jobs to machines while minimizing the maximum job processing time in the system. These models account for process uncertainty, allowing for robust scheduling solutions.

The multi-objective approach of using a sum or linear combination of weights is commonly applied to solve the P-MOFfJSP problem. This technique was first proposed by Zadeh (1963), who demonstrated that solving a scalar optimization problem with the objective function as a weighted sum of the components of the original vector function can lead to efficient solutions. Reducing the problem to a single objective function allows for comparing all alternatives, enabling a comprehensive order framework. Hence, the choice of weight values α_i plays a crucial role in achieving the final decision. It is essential to perceive how this choice impacts the optimal points in determining the coefficients.

One of the main advantages of this method is its computational efficiency and ease of use. However, the difficulty of determining appropriate weighting coefficients without adequate information is a significant drawback. Properly scaling objectives also require considerable knowledge about the problem, which may not always be available and costly. Additionally, this method may fail to generate certain parts of the Pareto front when its shape is concave, irrespective of the weight combinations. Nonetheless, incorporating additional functions may prove helpful in obtaining an initial sketch of the Pareto front for a given problem or providing background information to be used by another approach (Chiandussi et al. 2012; Al-Janabi and Alkaimi 2020, 2022; Al-Janabi et al. 2020a, 2020b; Al-Janabi et al. 2021; Mohammed and Al-Janabi 2022). Additionally, TLBO has been used to solve

supply chain problems such as inventory optimization and supplier selection (Baykasoğlu et al. 2014) and to solve transportation problems such as vehicle routing and scheduling (Jin et al. 2021). Furthermore, TLBO has shown promising results in solving complex engineering problems, such as design of truss structures and optimization of renewable energy systems (Mane et al. 2020). The versatility of TLBO in solving various optimization problems demonstrates its potential to be used in various fields. Overall, TLBO has proven to be a valuable tool in solving complex optimization problems with high efficiency and accuracy (Zou et al. 2019). For example, Xu et al. (2022) proposed a new variant of the Teaching–Learning-Based Optimization (TLBO) algorithm named Distance-Fitness Learning TLBO (DFL-TLBO). This variant employs a distance-fitness learning strategy to enhance search ability and address premature convergence and local optima entrapment issues in complex optimization problems.

The proposed methodology combines the Teaching–Learning-based Optimization (TLBO) algorithm with a Genetic Algorithm to solve the Multi-Objective Flexible Fuzzy Job Shop problem with partial flexibility (P-MOFfJSP). The Adaptive TLBO (TLBO-A) algorithm addresses the challenges associated with production scheduling in fuzzy time environments incorporating two genetic operators (mutation and crossover) with an adaptive strategy for population reconfiguration. This methodology offers several advantages over traditional metaheuristic algorithms, like genetic algorithms (Katoch et al. 2021). Firstly, TLBO-A enhances solution space exploration, ensuring the search for good solutions and avoiding premature convergence. Secondly, the hybridization of TLBO-A and GA promotes the synergy of these methods, exploiting the best of both techniques. Finally, the TLBO-A algorithm's adaptive strategy for population reconfiguration provides the ability to adapt to changes in the solution landscape, offering robustness to the algorithm.

The experimental evaluation of the proposed methodology on benchmark instances commonly used in programming problems with fuzzy variables demonstrates significant results, indicating that the TLBO-A algorithm outperforms among other state-of-the-art metaheuristic algorithms. The performance of this methodology exceeds other metaheuristic algorithms in terms of both solution quality and computational efficiency. Our methodology can effectively solve real-world production scheduling problems, such as delivery and availability times, incorporating fuzzy processing and delivery times. As a whole, our contribution lies in addressing the challenges associated with production scheduling in fuzzy time environments and proposing a practical hybrid metaheuristic approach.

3 Description of the problem

3.1 Multi-objective fuzzy flexible job shop scheduling problem with partial flexibility

Most research studies on scheduling problems focus on optimization methods to minimize makespan. However, processing times are subject to variations due to different situations. Hence, the makespan and variables associated with processing time are expected to fluctuate (Pan et al. 2014; Joo et al. 2018; Jin et al. 2021). Some studies have considered multi-objective optimizations in scheduling problems, but they have mainly worked with deterministic processing times. Therefore, the existing optimization techniques are unsuitable for solving such problems, as actual start or finish times will oscillate and create significant deviations between the actual and planned makespan. Recent research (Jin et al. 2016) has emphasized the need for optimization techniques that can handle stochastic processing times to address this issue.

Petrović et al. (2014) proposed fuzzy logic to handle imprecise information by combining fuzzy sets according to predetermined rules to generate various output values. This approach can effectively model uncertainty in real-world environments. To better represent the uncertainty inherent in the Flexible Job Shop Problem (FJSP), this study employs fuzzy sets to represent processing time and related variables like the start and end times of operations and jobs, machine utilization time, program completion time, and job processing time. It should be noted that in practical situations, the processing time for each job and operation is estimated based on the nature of the machine or the variability of personnel involved.

The multi-objective fuzzy flexible job shop scheduling problem with partial flexibility (P-MOfFJSP) consists of a set of the n jobs $J_i(i = 1, 2, 3...n)$ and a set of the m machines $M_K(k = 1, 2, 3...m)$. Each job J_i consists of n_i Operations. Several machines can process each operation O_{ij} that indicates the operation j of job$i$. The processing time of the operation O_{ij} on M_K is represented as a Triangular Fuzzy Number (TFN), $P_{ijk} = \left(\alpha^1_{ijk}, \alpha^2_{ijk}, \alpha^3_{ijk}\right)$ (Zheng et al. 2012).The P-MOfFJSP is characterized by a finite set of machines $M = \{M_1, M_2, M_3...M_m\}$ with heterogeneous characteristics and must process n jobs $J = \{J_1, J_2, J_3...J_n\}$, each job J_i is composed of n_i operations $O = \left\{O_{i,1}, O_{i,2}, O_{i,3}...O_{i,n_i}\right\}$, the total number of operations $N = \sum_{i=1}^{n} n_i$. The processing time of $O_{i,j}$ on the machine m_k is represented by TFN, $p_{ijk} = \left(p^1_{ijk}, p^2_{ijk}, p^3_{ijk}\right)$, where p^1_{ijk}, p^2_{ijk} and p^3_{ijk} represent the best, most likely, and worst processing times.

The transfer and setup time of the jobs between the different machines is included in the processing times. The fuzzy makespan of $O_{i,j}$ is represented by a TFN, $\widetilde{C}_{ijk} = \left(C^1_{ijk}, C^2_{ijk}, C^3_{ijk}\right)$, and the fuzzy total flow time of the schedule for $\widetilde{FT} = \left(FT^1, FT^2, FT^3\right)$, where C^1_{ijk} andFT^1, C^2_{ijk} andFT^2, and C^3_{ijk} and FT^3, respectively, represent the best, most probable and worst makespan and total flow time. The P-MOFfJSP aims to determine the allocation of machines and the sequencing of operations on all machines to minimize fuzzy makespan: $\widetilde{C}_{\text{Max}} = \max_{i=1,2,3,\ldots,n} \widetilde{C}_i$, and fuzzy average flow time: $\overline{F} = \frac{1}{n}\sum_{i=1}^{n} \widetilde{F}_i$, in which $\widetilde{C}_i$ is the fuzzy makespan of job J_i and $\widetilde{F}_i$ is the fuzzy flow time of jobJ_i, subject to precedence and processing constraints.

- All jobs are available at time zero (0).
- All machines are available at time zero (0).
- Each job can only be executed by one machine at a time.
- Jobs are independent of each other.
- Jobs consist of dependent operations that cannot be interrupted. Each operation must be completed before starting another one.
- Machines are independent (Flexible Job Shop, Heterogeneous Machines).
- At any given time, a machine can only run one job and become available for other jobs once the current job is finished.
- The processing times of jobs exhibit fuzzy behavior and are studied within a fuzzy interval defined by a fuzzy triangular number.
- The flexibility of the system is partial, with not all machines capable of processing all job operations.
- Setup times between jobs are included in processing times.
- The completion times of jobs also exhibit fuzzy behavior.
- Jobs are considered finished once all operations are executed.

The processing restrictions proposed in this system allow for flexible modeling of production systems, considering changes in job assignments and processing times. This increases efficiency and adaptability in response to changing customer requirements, leading to improved customer loyalty, cost reduction, and high-quality product guarantees. The fuzzy makespan is defined as $\widetilde{C}_{\max} = \max_{i=1,\ldots,n} \widetilde{C_i}, i = 1, 2, 3\ldots n$, where $\widetilde{C_i} = \left(c_i^1, c_i^2, c_i^3\right)$ is the fuzzy completion time of job i. The fuzzy average flow $(\widetilde{\text{FP}})$ is given by the expression $\widetilde{\text{FP}} = (F^1 + 2F^2 + F^3)/(4 * n)$. The P-MOfFJSP involves fuzzy number operations: addition, maximum, and ranking operations to establish the objective value and the feasible schedule. Given two fuzzy triangular numbers, $X = (x_1, x_2, x_3)$, $Y = (y_1, y_2, y_3)$, the operation addition is expressed as$X + Y = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$.

For the ranking operation, different criteria are used to compare:

- Compare the value of $G_1(X) = (x_1 + 2x_2 + x_3)/4$ and $G_1(Y) = (y_1 + 2y_2 + y_3)/4$. If $G_1(X) > G_1(Y)$, then $X > Y$.
- If both have the same value of G_1, then compare the value $G_2(X) = x_2$ and $G_2(Y) = y_2$. If $G_2(X) > G_2(Y)$, then $X > Y$.
- If G_1 and G_2 are identical, then compare the value of $G_3(X) = x_3 - x_1$ and $G_3\ (Y) = y_3 - y_1$. If $G_3(X) > G_3(Y)$, then $X > Y$.
- In the approximate max operation, if $X > Y$, then $X \bigvee Y = X$, else $X \bigvee Y = Y$.

In this formulation, to obtain the completion time of an operation based on its start time and processing time, the max operation is utilized. Additionally, to determine the fuzzy start time of a job operation on a specific machine M_k, the "max" operation is employed, considering the completion time of the previous operation in the job and the completion time of the previous operation in any job developed on M_k. Finally, the optimal schedule for the fuzzy problem is obtained using the "ranking" method to determine the order of completion time.

In this study, to structure the mathematical model, we referred to the variable-based precedence model proposed by Özgüven et al. (2010) for FJSP and the approach developed by Li and Gao (2016) for fFJSP, which are adapted to the proposed multi-objective model that considers the uncertainty of the system processes through fuzzy processing times. The problem considers partial flexibility in job assignments and heterogeneous characteristics of the available machines at each workstation. The sets, parameters, and equations are defined as follows.

3.1.1 Sets

J: Set of jobs;$J = \{J_1, J_2, J_3 \ldots J_n\}$.$i, h$: Index of the job; $i, h = 1, 2, 3\ldots n$.

M: Set of machines; $M = \{M_1, M_2, M_3 \ldots M_k \ldots M_m\}$.$k$: Index of machine; $k = 1, 2, 3 \ldots m$.

O: Set of operations of the job i; $O = \left\{O_{i,1}, O_{i,2}, O_{i,3} \ldots O_{i,n_i}\right\}$.

O_{ij}: Operation j of job i.j, g: Index of operation; $j, g = 1, 2, 3\ldots n_i$.

3.1.2 Parameters

n: Number of jobs. m: Number of machines. n_i: Total numbers of operations of job i.

M_{ij}: Set of machines available for operation; O_{ij} $\left(M_{ij} \subseteq M_k\right)$.$\widetilde{p}_{ijk}$: Fuzzy processing time of the O_{ij} on the machine k; $(\widetilde{p}_{ijk} \geq (0,0,0))$.$\beta$: Relative importance of maximum fuzzy completion time on the schedule.

$1-\beta$: Relative importance of fuzzy average flow time on the schedule.

B: A large number.

3.1.3 Decision variables

$$x_{ijk} = \begin{cases} 1 & \text{if the machine } k \text{ is selected for the run } O_{ij} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ijk} = \begin{cases} 1 & \text{if the operation } O_{ij} \text{ precedes operation } O_{hg} \text{ on machine } k \\ 0 & \text{otherwise} \end{cases}$$

$\widetilde{t}_{ijk}$: Fuzzy start time of the O_{ij} if run on the machine M_k

$\widetilde{C}_{ijk}$: Fuzzy completion time of the O_{ij} on the machine M_k.

$\widetilde{C}_i$: Fuzzy completion time of job i.

$\widetilde{C}_{max}$: Maximum fuzzy completion time of jobs.

$\widetilde{F}_i$: Fuzzy flow time of job$i$.

$\widetilde{FT}$: Fuzzy total flow time of schedule.

$\widetilde{\mathrm{FP}}$: Fuzzy average flow time of schedule.

The mathematical model of the MOFfJSP of the research is:

$$\text{Min } Z_1 = \widetilde{C}_{\max} \tag{1}$$

$$\text{Min } Z_2 = \widetilde{\mathrm{FP}} \tag{2}$$

$$\text{Min } Z_3 = \beta Z_1 + (1-\beta) Z_2 \tag{3}$$

S.t:

$$\widetilde{C}_{\max} \geq \widetilde{C}_i; \forall (i) \tag{1}$$

$$\widetilde{C}_i \geq \sum_{k \in M_{i,j}} \widetilde{C}_{i,j,k}; \forall (i, j = J_{n_i}, k) \tag{2}$$

$$\widetilde{t}_{i,j,k} + \widetilde{C}_{i,j,k} \leq x_{i,j,k} * B; \forall \left(i,j,k \in M_{ij}\right) \tag{3}$$

$$\widetilde{C}_{i,j,k} \geq \widetilde{t}_{ijk} + \widetilde{p}_{i,j,k} - \left(1 - x_{i,j,k}\right) * B; \forall \left(i,j,k \in M_{ij}\right) \tag{4}$$

$$\widetilde{t}_{i,j,k} \geq \widetilde{C}_{h,g,k} - y_{i,j,h,g,k} * B; \forall \left(i \leq h, j, g, k \in M_{ij} \cap M_{hg}\right) \tag{5}$$

$$\widetilde{t}_{h,g,k} \geq \widetilde{C}_{i,j,k} - \left(1 - y_{i,j,h,g,k}\right) * B; \forall \left(i \leq h, j, g, k \in M_{ij} \cap M_{hg}\right) \tag{6}$$

$$\sum_{k \in M_{i,j}} \widetilde{t}_{i,j,k} \geq \sum_{k \in M_{i,j}} \widetilde{C}_{i,j-1,k}; \forall \ (i, j = 2, \ldots, J_{n_i}) \tag{7}$$

$$\sum_{k \in M_{i,j}} x_{i,j,k} = 1; \forall (i,j) \tag{8}$$

$$\widetilde{F}_i = \sum_{i=1}^{n} \widetilde{C}_{i,j,k}; \forall (j,k) \tag{9}$$

$$\widetilde{FT} = \sum_{i=1}^{n} \widetilde{F}_i \tag{10}$$

$$\widetilde{\mathrm{FP}} = \frac{1}{n} \sum_{i=1}^{n} \widetilde{F}_i \tag{11}$$

$$\widetilde{t}_{i,j,k} \geq 0, \widetilde{C}_{i,j,k} \geq 0; \forall (i,j,k) \tag{12}$$

$$\widetilde{C}_i \geq 0; \forall (i) \tag{13}$$

$$x_{i,j,k} \in \{0,1\}; \forall (i,j,k) \tag{14}$$

$$y_{i,j,h,g,k} \in \{0,1\}; \forall \left(i \leq h, j, g, k \in M_{ij} \cap M_{hg}\right) \tag{15}$$

The proposed model is formulated through constraints to optimize the flexible job shop problem with fuzzy processing times. Constraint (1) establishes the maximum completion time (fuzzy makespan) of the jobs, while Constraint (2) determines the fuzzy completion time of the final operations of the jobs. To ensure proper assignment, Constraint (3) sets the start and end fuzzy times on machine M_k to zero when the operation O_{ij} is not assigned to it. Constraint (4) guarantees that the difference between the fuzzy starting and completion times is equal to, at least, the fuzzy processing time on machine M_k. Constraints (5) and (6) maintain the requirements that the operation O_{ij} and O_{hg} cannot be conducted simultaneously on any machine in the set $M_{ij} \cap M_{hg}$. To avoid violation of precedence relationships, Constraint (7) ensures that the operations of a job are developed in the correct order. Similarly, Constraint (8) confirms that an operation is only performed on a single machine. Constraint (9) calculates the fuzzy flow time of the jobs, and Constraint (10) determines the fuzzy total flow time of the schedule. Furthermore, Constraint (11) establishes the fuzzy average flow time of the schedule. Constraints (12) and (13) secure the nonnegativity of the variables. Constraints (14) and (15) define the binary variables. Finally, to include all scenarios, the TFN values are considered individually to represent the best, most probable, and worst outcomes.

4 Optimization algorithm based on adaptive teaching–learning

Teaching Learning-Based Optimization (TLBO) is a metaheuristic algorithm inspired by the classroom teaching–learning process and individuals' social behavior (Baykasoğlu et al. 2014). Rao proposed a relatively new method that has gained widespread acceptance in solving engineering problems. The standard TLBO model involves the influence of teachers on students' learning levels during a training process, with the results being evaluated in terms of grades. In TLBO, the teacher is considered the most knowledgeable individual in the class (population), which helps improve the overall knowledge level according to their ability. The students' quality is assessed based on the mean value of the population. Initially, TLBO was designed for single-objective nonlinear optimization problems, but it has since proven effective for general engineering problems (Pickard et al. 2016). According to the study by Rao and Rai (2016), TLBO's effectiveness has been extensively verified with various performance criteria, demonstrating superior performance over other methods with less computational effort, particularly for large-scale problems.

Baykasoğlu et al. (2014) and Xu et al. (2015) conducted a study evaluating and analyzing TLBO's performance in solving two well-studied combinatorial optimization problems in the literature: the Flow Shop Problem and the Flexible Job Shop Problem. The study demonstrated the efficiency of TLBO when compared to other metaheuristics. The authors conducted also a literary review to understand the fundamental characteristics of these metaheuristics.

Several authors have proposed modifications and additional stages to TLBO to enhance its efficiency. The improved TLBO algorithm includes population division into multiple groups, training tutorials, and self-motivating learning operators (Yu et al. 2018). These modifications can be categorized into techniques used to generate the initial population, which includes learning strategies, adaptive parameters, genetic operators, neighborhood management, and population diversification (Zou et al. 2019).

Opposition-Based Learning (OBL) and the Nawaz–Encore–Ham method have been proposed as alternative methods for generating the initial population in TLBO by Shao et al. (2017). Mandal and Roy (2013) and Roy and Sarkar (2014) proposed a modification of the OBL called Quasi—Opposition-Based Learning. The adaptive parameters have been designed for TLBO. Satapathy et al. (2013) presented an adaptive weight parameter that decreases linearly in the teaching and learning phase according to the number of iterations. This study added the differential weight vector, inertial weight and acceleration coefficient, while Bulbul and Roy (2014) proposed a new TLBO method with an adaptive dynamic control mechanism for the nonlinear economic load dispatch problem.

The combining of TLBO-A and an evolutionary algorithm presents a promising approach for achieving better-quality solutions in a shorter computational time. This is supported by previous literature highlighting both metaheuristics' efficiency in solving complex optimization problems. By implementing the adaptive phase approach (as shown in Figs. 1 and 4), the parameterization of feasible solutions can be fine-tuned to produce more effective results reducing the computational burden. The application of evolutionary operators in TLBO-A can further enhance the search for optimal solutions. Overall, this integration of TLBO-A and evolutionary algorithms offers a powerful optimization tool that can provide high-quality solutions in less time.

The optimization algorithm based on Teaching–Learning-Based Optimization (TLBO) is a powerful and efficient heuristic for solving complex optimization problems. TLBO incorporates a particular encoding scheme to represent solutions and a decoding method to transfer solutions to a feasible schedule in a fuzzy sense (Baykasoğlu et al. 2014). The algorithm also integrates a two-phase crossover scheme based on the teaching and learning mechanism and special local search operators to balance exploration and exploitation capabilities. TLBO has been compared with other existing algorithms and demonstrated superior effectiveness and efficiency. In particular, TLBO can find high-quality solutions within a reasonable time, making it competitive in speed and efficiency. The TLBO can also find optimal or near-optimal solutions in most test instances, making it a competitive method in accuracy and convergence (Mane et al. 2020). In summary, TLBO is a competitive heuristic for solving complex optimization problems due to its potential for finding high-quality solutions in a reasonable time and its ability to converge to optimal or near-optimal solutions in most test instances (Jin et al. 2021).

The proposed optimization algorithm based on Teaching Learning (TLBO) is a valuable solution to solve the complex Flexible Job Shop Scheduling problem with fuzzy processing time. A specific coding scheme is always used to represent feasible solutions, eliminating the need for an additional method to decode and transfer solutions to a feasible program based on the principles of fuzzy logic.

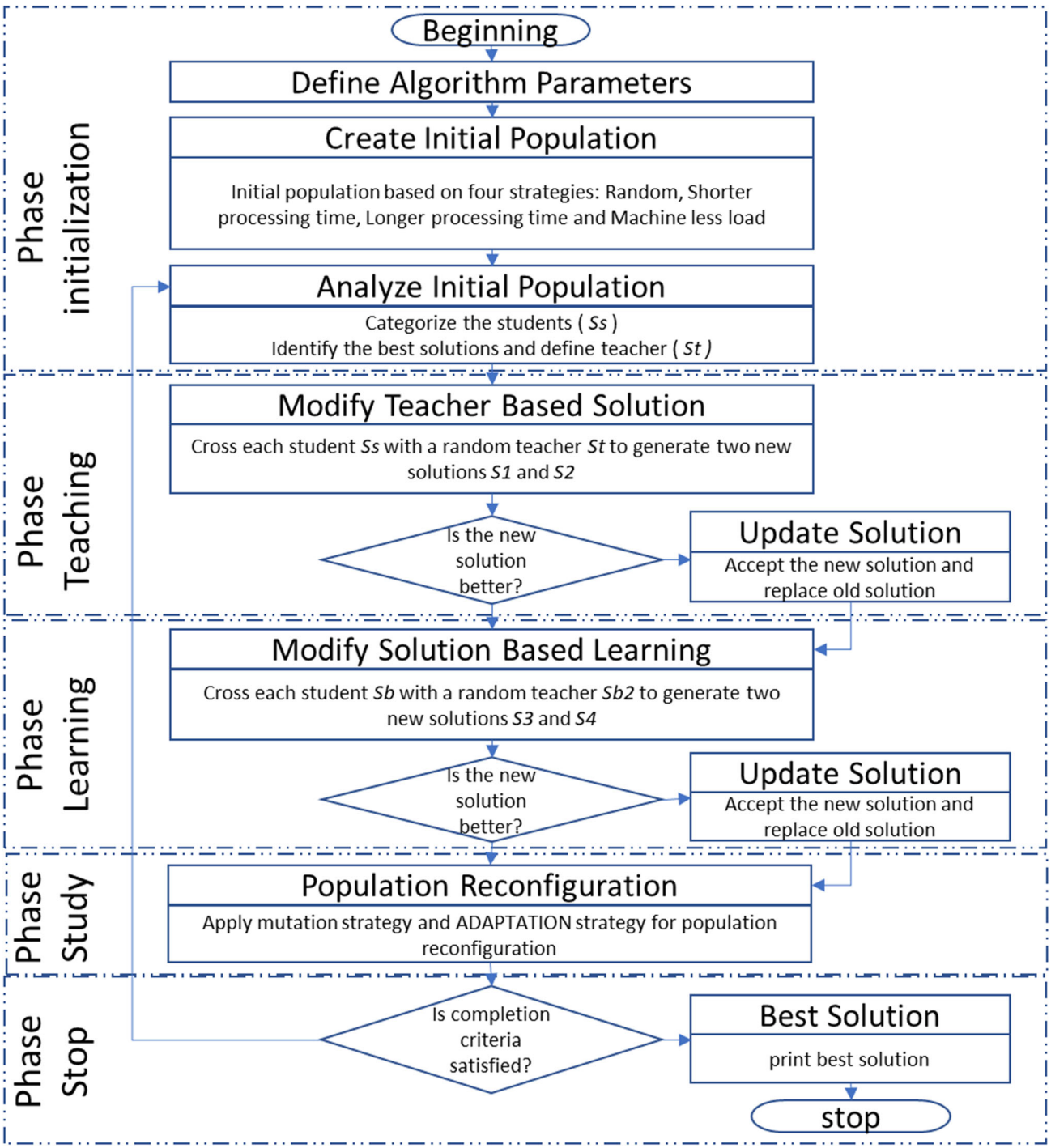


Fig. 1 Evolutionary TLBO phases proposed

Two crossing schemes (Bi-Point, POX) and mutation were integrated through two strategies (Sequence Vector Mutation, Machine Assignment Vector Mutation) to ensure a balance between exploration and exploitation while searching for reasonable solutions. Moreover, a novel adaptation strategy is incorporated to reconfigure a portion of the population by reassigning operations to machines with less load assignment when searching for better solutions is trapped in a local optimum. The TLBO-A algorithm is also developed in a multi-objective environment, making it versatile and suitable for systems with partial or total flexibility within the work routes and available system resources. These features make it a better algorithm for developing programming schemes.

According to Ji et al. (2017), the standard TLBO algorithm is prone to get stuck in local optima when the population's diversity is lost, posing a significant problem. To address this issue, the TLBO-A algorithm is developed and experimentally tested. The tests run for single- and multi-objective problems, allowing for comparability and the algorithm's efficiency validation. In conclusion, TLBO-A is a hybrid improvement metaheuristic incorporating genetic operators to optimize non-polynomial problems such as Multi-objective Flexible Job Shop Scheduling with fuzzy processing times. TLBO-A includes two new stages: the Study Phase and the Adaptation Phase. The algorithm compares and optimizes the makespan and uncertain average flow time on each machine to obtain the most promising solution while protecting against uncertainty by defining different levels of fluctuations in completion times.

4.1 Adaptive teaching–learning-based optimization algorithm (TLBO-A)

A new adaptive strategy is proposed below to enhance the search process for optimal solutions and increase the likelihood of promising results during its evolution. This adaptive strategy is designed to improve the global and local search capacity by exploring new scenarios that provide the algorithm with greater diversity, thus preventing it from being trapped in suboptimal solutions due to premature convergence. The proposed Adaptive Teaching–Learning-based Optimization Algorithm (TLBO-A) builds on the basic principles of TLBO, incorporating two genetic operators (mutation and crossover) and an adaptive strategy for population reconfiguration, ensuring enhanced exploration of the solution space and avoidance of premature convergence. In TLBO-A, each member of the population is considered a student, representing a potential solution to the problem. A solution comprises a sequence of operation vectors and a machine assignment vector.

The sequence vector in the proposed algorithm represents the order in which operations of the different jobs are executed, and its length is equal to the total number of operations. The job index indicates the operation and its position in the sequence vector. The operations of each job are arranged in ascending order in the sequence vector. On the other hand, the machine allocation vector denotes the indices of the machines selected to process each job operation, as shown in Fig. 2.

According to Fig. 2, operation one of job one (O_{11}) will be processed on machine four (M_4). Operation two of job one (O_{12}) will be processed on machine three (M_3). Operation three of job one (O_{13}) on machine one (M_1) and job one operation four (O_{14}) on machine two (M_2).

The TLBO-A is characterized by four key parameters, namely the population size (i.e., the number of students), the number of learning levels (i.e., Levels), the percentage of top-performing students or teachers (i.e., Alpha), and the mutation percentage. In the initial phase of the algorithm, the parameters, population, and Beta parameter, which corresponds to the objective related to the makespan, are initialized. The individuals are then sorted into different learning levels, with the top group comprising the Alpha expression, representing the best students or teachers within the population. To generate the initial population, the TLBO-A algorithm considers several strategies. The sequence vector is randomly generated, while the machine allocation vector is created using different rules, each with an associated probability of occurrence. Specifically, Rule 1 involves the random selection of a machine for each operation. In Rule 2, the machines with the shortest processing time are selected from the available machines for each operation. Conversely, in Rule 3, the machines with the longest processing time are selected from the available machines for each operation. Lastly, in Rule 4, the machines with the minor load are chosen from the available machines.

Loss of population diversity can cause the traditional TLBO algorithm to fall into local optima (Ji et al. 2017). The proposed TLBO algorithm incorporates crossing and mutation operators to address this issue and enhance search space exploration. Genetic algorithms commonly use these basic operators to increase population diversity and prevent premature convergence. Typical forms of crossing include PMX (Partially Matched Crossover), OX (Ordered Crossover), and CX (Cycle Crossover) (Acevedo Chedid et al.

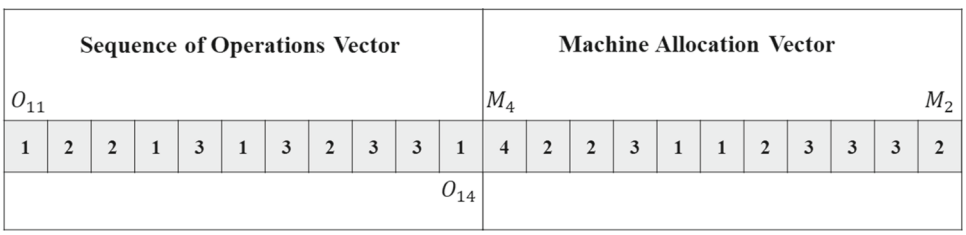


Fig. 2 Representation of a feasible solution

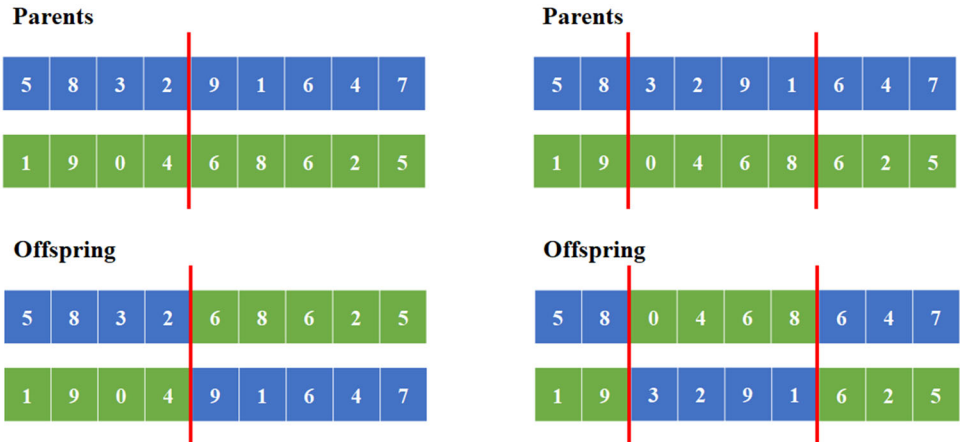


Fig. 3 Example of crossing a point and bi-point

2020). The two-point crossing method is employed at this stage, as depicted in Fig. 3.

In the proposed TLBO, the Teaching phase involves crossing each student in the population outside the top group with a randomly selected teacher to generate two new solutions using the POX crossing strategy. The first generated student is then compared to the student selected for crossing, and if the new solution is better, it replaces the previous student; otherwise, it is maintained. The exact process is repeated for the second generated student and the selected teacher for crossing. In the Learning phase, each teacher in the population crosses paths with another teacher to find a new solution using the Bi-point Crossing strategy. If the resulting solution has a better learning degree (higher fitness value), the initial teachers are replaced within the population. Otherwise, the selected individual is maintained.

The Study phase of the TLBO-A algorithm involves applying the mutation operator to the population's students to enhance their learning degree (Fitness Value). This process employs both sequence mutation strategy and machine allocation mutation strategy. The machine allocation vector mutation uses three criteria: selection of the machine with the shortest time, random selection, and selection from a machine with a higher load to one with a lower load. In the Adaptive phase of the TLBO-A algorithm, an adaptive strategy is implemented to reconfigure a percentage of the current population. This strategy applies a change criterion when the best result (the best teacher) is repeatedly obtained at certain levels. The students are ordered based on their learning degrees during this phase. Finally, the TLBO-A checks for compliance with the stop criteria. The TLBO-A metaheuristic's structure is explained in Fig. 4.

Using the Java programming language, the TLBO-A algorithm was implemented in NetBeans IDE 8.2. The corresponding pseudocodes for each stage are provided in the annexes, which detail the algorithm coding process. Appendices 1–22 specifically list the coding process for the main class, data entry, initial population generation, sequence mutation operator, assignment mutation operator, adaptive strategy, and convergence graph generation method.

The proposed Adaptive Teaching–Learning-based Optimization Algorithm (TLBO-A) is a novel approach that enhances the global and local search capacity by incorporating two genetic operators (mutation and crossover) and an adaptive strategy for population reconfiguration. The TLBO-A algorithm is based on the basic principles of TLBO, but it includes a sequence vector and a machine allocation vector to represent the order in which operations of different jobs are executed and the indices of the machines selected to process each job operation. Using these vectors in combination with the proposed adaptive strategy provides a more diverse search space and avoids premature convergence, resulting in better and more promising solutions. The TLBO-A algorithm is an effective and versatile optimization method used in multi-objective scheduling problems with fuzzy processing times, making it an ideal choice for developing programming schemes.

4.2 Validation of TLBO-A metaheuristic

For the validation stage of the TLBO-A algorithm, five were utilized instances developed by Lei (2010), each containing data on fifteen (15) jobs, ten (10) machines, and eighty (80) operations. Table 2 provides a summary of the parameters for each instance.

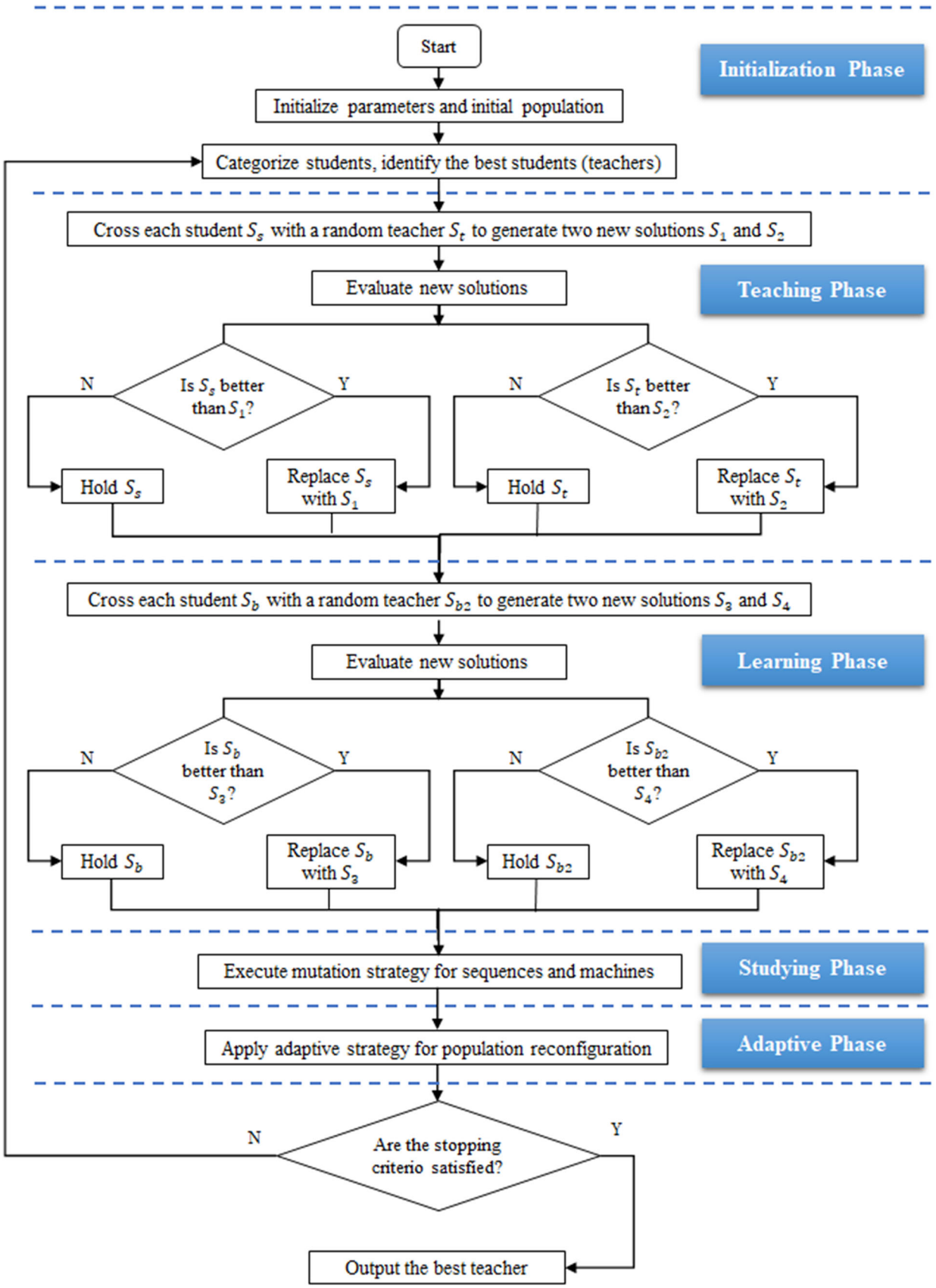


Fig. 4 The framework of the TLBO-A for the FJSPF

Table 2 Details of the instances with total flexibility by Lei (2010)

Instance	Number of jobs	Number of machines	Total operations
1	10	10	40
2	10	10	40
3	10	10	50
4	10	10	50
5	15	10	80

Table 3 Experimental design parameters for FfJSP

Parameters	Factor level			
	1	2	3	4
Population	500	1000	1500	2000
Generation	500	1000	1500	2000
P. mutation	0.2	0.4	0.6	0.8
Alpha	0.2	0.4	0.6	0.8

The TLBO-A algorithm is applied to minimize the fuzzy completion time (fuzzy makespan) to compare its performance to other existing algorithms. Subsequently, an experimental design is developed to analyze the execution of TLBO-A in solving the P-MOFfJSP. The design is able to establish the influence of each parameter on the proposed algorithm through the use of an orthogonal design of Taguchi L16 (4^4), which is applied to the experimental test data of instance two (2) by Lei (2010). Table 3 summarizes the parameters and corresponding levels for the Taguchi L16 design.

Table 3 shows the parameters and their respective levels. In each combination of parameters, 16 replicates are considered and calculated with the expression 2^k where k represents the factors ($k = 4$), a sample of 272 data is used in the design. The fuzzy makespan response variable is defined as $\frac{(x_1+2x_2+x_3)}{4}$ which corresponds to the weighted weight of the fuzzy triangular number.

The design of experiments is developed using Statgraphics software, and the results indicated that Population, Alpha, and Levels factors are significant, with P-values less than 5% (refer to Annexure 8). The population size is the most significant factor, suggesting that a large population size allows the TLBO-A to converge slowly and achieve better results. The Alpha factor is determined to be the second most significant factor. The levels factor is also found to be significant, and it is recommended to assign large values. The mutation factor has small impact on the algorithm's functioning; higher or lower intensity values do not affect the response variable. The parameters

Table 4 Algorithms to minimize fuzzy makespan

Algorithm	References
BSHH	Lin (2019)
HMVO	Lin et al. (2019)
HABC	Li et al. (2017)
HBBO	Lin and Zhang (2016)
DHS	Gao et al. (2016a, b)
FEDA	Liu et al. (2015)
TLBO	Xu et al. (2015)
EDA	Wang et al. (2012a, b)
CGA	Lei (2012)
BSA	Civicioglu (2013)
DIGA	Lei (2010)

established for the TLBO-A while minimizing the fuzzy makespan are: Population: 2000; Generations: 2000; P Mutation: 0.2; Alpha: 0.8.

The size of the population and the levels are the most significant parameters for the TLBO-A applied to the P-MOFfJSP. The Alpha factor is also essential for generating the solution, even though it is insignificant. For each parameter combination, 16 replications are conducted using a sample of 272 data. The response variables are defined as Fuzzy makespan: $\frac{(x_1,+2x_2+x_3)}{4}$; Fuzzy average flow $\frac{(y_1,+2y_2+y_3)}{(4*\text{Numberofjobs})}$. Where X is a fuzzy triangular number $X = (x_1, x_2, x_3)$ that represents the fuzzy completion time associated with the job, and Y is another fuzzy triangular number $Y = (y_1, y_2, y_3)$ that represents the average fuzzy flow time.

Assigning large values to the population and level factors is one way to avoid premature convergence. The behavior data of the diffuse makespan response variable and diffuse average flow concerning each factor can be observed in Annex 9, which displays the standardized effect of each factor for the fuzzy makespan response variable and the diffuse average flow response variable. Using the statistical software Statgraphics, the values that should be assigned to the TLBO-A factors were determined to minimize the objectives simultaneously, expressed through the response variables. Based on the results obtained from the experiment design, the parameterized values for the P-MOFfJSP are Population: 2000; Generation: 2000; P. Mutation: 0.4; Alpha: 0.6.

5 Experimental results

The TLBO-A is executed in 18 independent runs using the instances provided by Lei (2010) and Lin (2002), with the parameterized factors as the input. For the Total Flexibility

Table 5 Results and comparisons of the instances for the fuzzy makespan

Instance	Algorithm	Middle value	Best value	Worst value
1	BSHH	**(18.5–26.9–36.0)**	**(18–26–36)**	**(18–27–37)**
	HMVO	(19.9–28.0–38.1)	(21–28–37)	(19–28–39)
	TLBO-A	(**20.97–30.68–39.22**)	(**19–28–38**)	(**21–32–41**)
	HABC	(21.0–32.0–43.6)	(19–30–43)	(23–33–46)
	HBBO	(20.8–28.0–37.2)	(21–28–37)	(19–28–39)
	DHS	(19.9–29.6–40.6	(17–29–39	(19–31–44)
	FEDA	(21.8–32.5–43.8)	(20–32–41)	(23–34–43)
	TLBO	(20.3–29.9–40.9	(19–28–39)	(21–32–42)
	EDA	(20.3–30.5–41.6)	(20–28–40)	(22–32–43)
	CGA	(23.1–33.1–43.4)	(21–29–41)	(25–37–47)
	BSA	(20.3–28.5–39.5)	(19–28–40)	(23–29–39)
	DIGA	(22.8–33.4–44.6)	(20–31–40)	(25–37–49)
2	BSHH	**(28.8–40.0–52.5)**	**(29–39–51)**	**(32–39–54)**
	HMVO	(30.0–45.0–58.0)	(30–45–58)	(30–45–58)
	TLBO-A	(**31.25–47.65–58.25**)	(**32–47–57**)	(**33–48–61**)
	HABC	(33.0–47.8–62.2)	(33–46–58)	(36–48–65)
	HBBO	(30.0–45.0–58.0)	(30–45–58)	(30–45–58)
	DHS	(32.1–46.2–59.3)	(30–45–61)	(35–46–62)
	FEDA	(35.7–48.4–59.8)	(35–46–57)	(36–50–61)
	TLBO	(32.6–46.4–58.5	(30–45–58)	(36–49–63)
	EDA	(33.7–46.9–57.9)	(32–46–57)	(34–48–58)
	CGA	(36.4–50.8–66.0)	(34–47–63)	(38–53–71)
	BSA	(31.6–44.7–58.8)	(30–45–58)	(34–44–59)
	DIGA	(35.4–48.4–62.3)	(33–48–57)	(37–50–65)
3	BSHH	**(29.5–42.5–55.9)**	**(30–42–54)**	**(28–44–56)**
	HMVO	(31.0–43.8–58)	(29–44–58)	(32–44–59)
	TLBO-A	(**33.81–47.38–59.7**)	(**29–45–59**)	(**33–51–67**)
	HABC	(33.9–50.8–67.3)	(33–47–64)	(36–54–70)
	HBBO	(30.2–43.8–58.1)	(30–42–60)	(31–45–57)
	DHS	(31.6–45.9–59.9)	(31–45–58)	(33–48–62)
	FEDA	(36.8–50-66.2)	(37–50-60)	(39–54-72)
	TLBO	(31.5–46.7–62.2)	(30–45-60)	(33–50–70)
	EDA	(32.8–47.2–62.9)	(31–46–60)	(34–49–66)
	BSA	(31.7–45.4–60.1)	(29–44–62)	(33–45–62)
	CGA	(36.4–50.8–66.0)	(34–47–63)	(38–53–71)
	DIGA	(37.3–53.0–66.9)	(37–49–64)	(41–58–75)
4	BSHH	**(21.5–33.0–46.3)**	**(21–32–47)**	**(24–33–46)**
	HMVO	(22.5–34.0–48.0	(25–33–47)	(25–34–48
	TLBO-A	(**24.25–38.343–53.375**)	(**25–37–50**)	(**25–39–56**)
	HABC	(25.5–40.0–56.3)	(23–38–53)	(25–44–59)
	HBBO	(22.8–34.0–47.9)	(24–33–47)	(23–35–48)
	DHS	(24.1–36.1–50.9)	(24–35–48)	(26–37–53)
	FEDA	(27.6–40.8–59.6)	(25–39-54)	(28–40-59)
	TLBO	(24.9–36.5–50.8)	(21–36-50)	(26–40–57)
	EDA	(32.8–47.2–62.9)	(31–46–60)	(34–49–66)
	CGA	(27.4–40.4–55.0)	(26–37–51)	(29–42–59)
	BSA	(24.1–35.4–49.1)	(26–34–48)	(25–36–50)
	DIGA	(29.2–42.9–57.5)	(29–41–56)	(29–46–60)

Table 5 continued

Instance	Algorithm	Middle value	Best value	Worst value
5	BSHH	**(35.3–52.6–73.0)**	**(36–52–69)**	**(33–53–77)**
	HMVO	(36.8–54.3–74.7)	(37–53–74)	(39–56–72)
	TLBO-A	(33.5–56.28–73.25)	(32–54–72)	(38–56–74)
	HABC			
	HBBO	(37.2–54.0–74.3)	(36–54–70)	(37–55–75)
	DHS	(37.9–55.8–77.8)	(36–54–74)	(42–59–84)
	FEDA	(46.2–64.4–90-6)	(48–65-88)	(48–66-93)
	TLBO	(36.1–57.5–78.2)	(36–55-72)	(37–61–82)
	EDA	(38.6–56.9–78.3)	(36–55–73)	(40–60–81)
	CGA	(47.0–65.4–86.0)	(42–62–82)	(49–70–91)
	BSA	(39.0–57.7–78.4)	(38–56–79)	(38–58–82)
	DIGA			

Numbers in bold correspond to the best results per instance

instances by Lei (2010), the TLBO-A is executed, and the results are compared with existing algorithms applied to the production scheduling problem in the FfJSP environment, considering the minimization of the Cmax fuzzy. Table 4 compares TLBO-A with other existing algorithms applied to the fuzzy flexible environment production scheduling problem, FfJSP, for each of the five instances.

The results are also presented in Table 5 alongside those obtained with the other algorithms. Table 5 shows the results, and comparisons of the instances for the fuzzy makespan obtained with the algorithms are also shown.

When applied to FfJSP, TLBO-A yields better results than the standard TLBO algorithm in Lei's (2010) instances, except for instances 2 and 5, where it does not outperform the BSHH algorithm. In comparing the results of our proposed heuristic TLBO-A to the BSHH algorithm, it is found that the BSHH algorithm produces the best results in all instances for the fuzzy makespan problem. However, the TLBO-A heuristic shows a variation in these best results ranging from 2 to 11% across all instances. When comparing the TLBO-A heuristic to the BSHH algorithm, it outperforms the latter in instances 1, 3, and 4, with an average fluctuation of only 1% compared to the TLBO heuristic.

The best solutions obtained with TLBO-A FfJSP for instances 2 and 5 are presented in Figs. 5 and 6, respectively. Figure 5 shows the best solution, for instance 2 compared with the TLBO-A FfJSP, including the programming of the jobs for each one of the ten considered machines. In the case of job 1, its four operations must be developed on machines M_6, M_7, M_3 and M_4, respectively. Job 2 is scheduled on machines M_3, M_6, M_9, and M_8. Job 3 operations 1, 2 and 4 are produced on machine M_{10} and operation three on machine M_5. According to the results presented in Fig. 5, the machine that develops the most jobs is M_1, with six jobs, followed by M_3, with five jobs and M_5, with four jobs.

Figure 6 presents the best solution for instance 5, compared with the TLBO-A FfJSP, including the programming of the jobs for each one of the ten considered machines. In the case of job 1, its five operations must be developed on machines M_2, M_{10}, M_2, M_1 and M_5, respectively. Job 2 is scheduled on machines M_8, M_8, M_7, M_2, M_6, and M_5. Job 3 operations 4 and 5 are produced on machine M_9, and operations 1, 2, and 3 on machines M_7, M_8, and M_3, respectively. According to the results presented in Fig. 6, the machine that develops the most jobs is M_1, with ten jobs, followed by M_5 and M_{10}, with nine jobs and M_3 and M_6, with eight jobs, respectively.

Figure 7 displays the TLBO-A algorithm's evolution for solving instance 2, showing the behavior of the fuzzy makespan objective against the fuzzy average flow and the Pareto frontier solutions (non-dominated solutions). The Pareto optimal solution is represented by the fuzzy triangular number (32–47–58) for the fuzzy makespan, with a weighted weight of 46.0 and a fuzzy average flow of 38.2. By varying the Beta parameter of the multicriteria function between zero (0) and one (1) with successive increments of 0.05, 21 possible solutions were obtained, resulting in the evolution presented in Fig. 7.

The P-MOFfJSP is solved by applying TLBO-A to the five instances provided by Lei (2010). Table 6 summarizes the general results obtained for each instance, including the mean, best, and worst values achieved for the considered

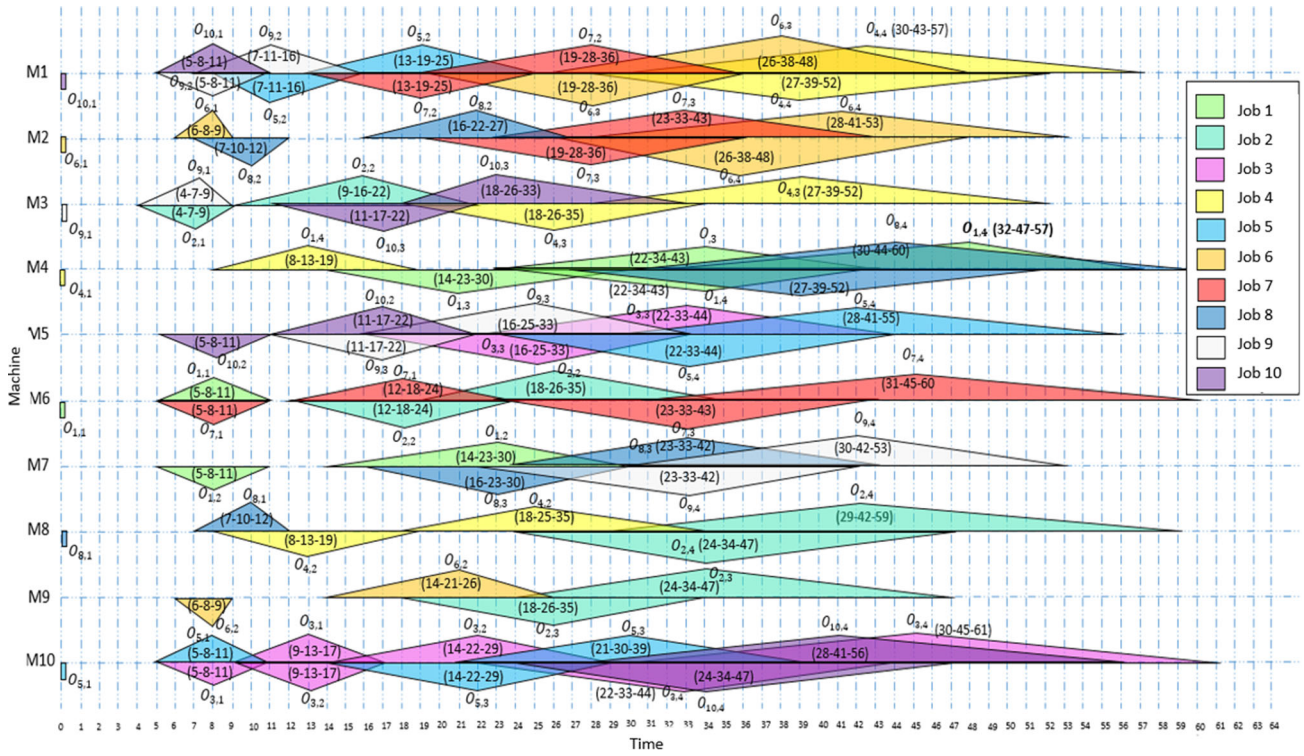


Fig. 5 The best solution for instance "2" (Lei (2010)) with the TLBO-A FfJSP

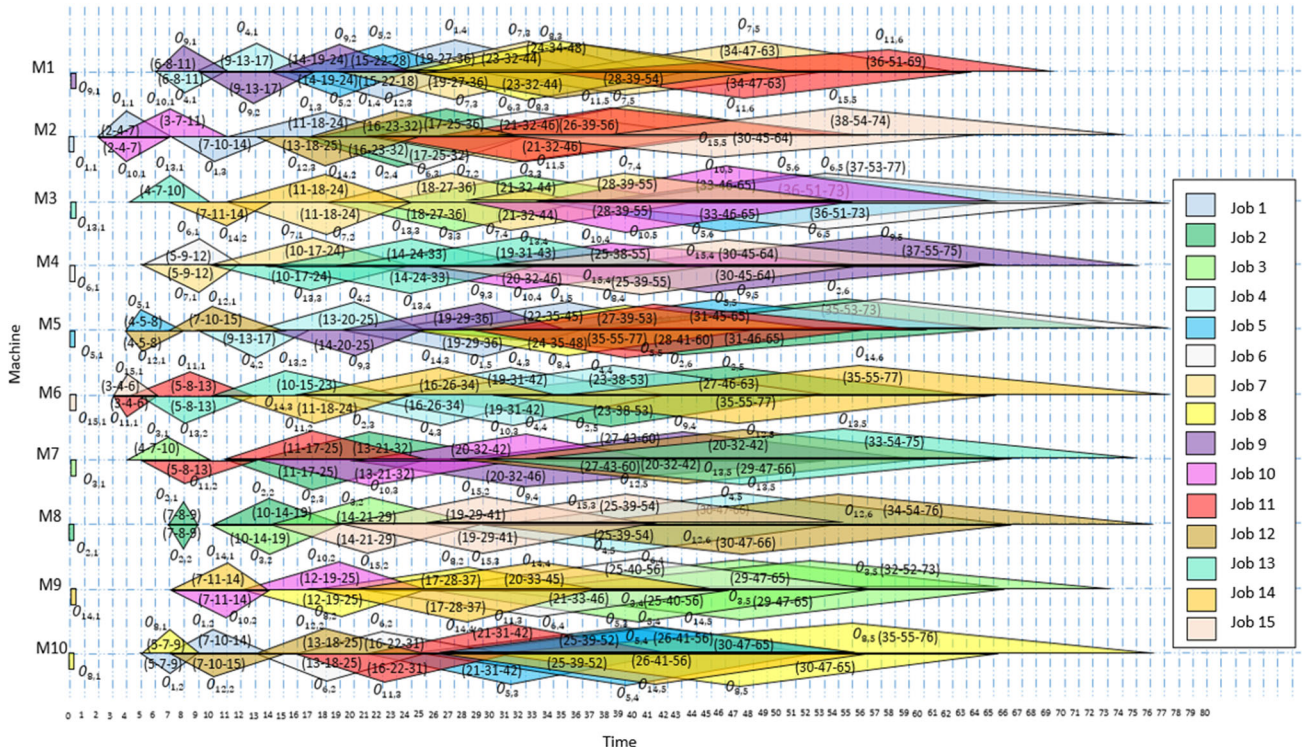


Fig. 6 The Best solution for instance "5" (Lei (2010)) with the TLBO-A FfJSP

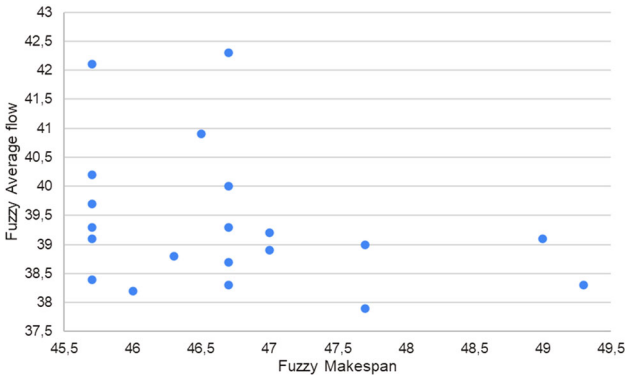


Fig. 7 Evolution of TLBO-A in the best solution for instance 2 (Lei (2010))

objectives. To evaluate each instance, 18 independent runs are executed. The Gantt charts with start and end times, the sequence of operations, and the operations assigned to each machine for the best results of instances 1 and 2 are presented in Figs. 8 and 9, respectively.

Figure 8 presents the best solution for instance 1, compared with the TLBO-A for the P-MOFfJSP, including the programming of the jobs for each one of the ten considered machines. In the case of job 1, its five operations must be developed on machines M_6, M_3, M_4, and M_1. Job 2 is scheduled on machines M_4, M_5, M_6, and M_9. Job 3 is scheduled on machines M_3, M_1, M_7, and M_2. According to the results presented in Fig. 8, the machine that develops the most jobs is M_1, with six jobs, followed by M_2, M_3, M_4,

Table 6 The result of the TLBO-A applied to the P-MOFfJSP

Instance	Fuzzy makespan			Fuzzy average flow		
	Middle value	Best value	Worst value	Middle value	Best value	Worst value
1	(21–30–41)	(19–29–40)	(21–30–41)	27.2	26.7	27.9
2	(31.6–46.8–57.1)	(32–47–58)	(32–47–57)	38.8	37.8	42.3
3	(32.5–47.8–63.8)	(31–45–62)	(34–48–77)	41.5	38.6	43.8
4	(27.8–38.2–55.2)	(24–38–55)	(29–39–57)	36.1	35.7	39.6
5	(35.7–57.5–74.3)	(32–55–73)	(36–59–75)	51.7	48.1	56.9

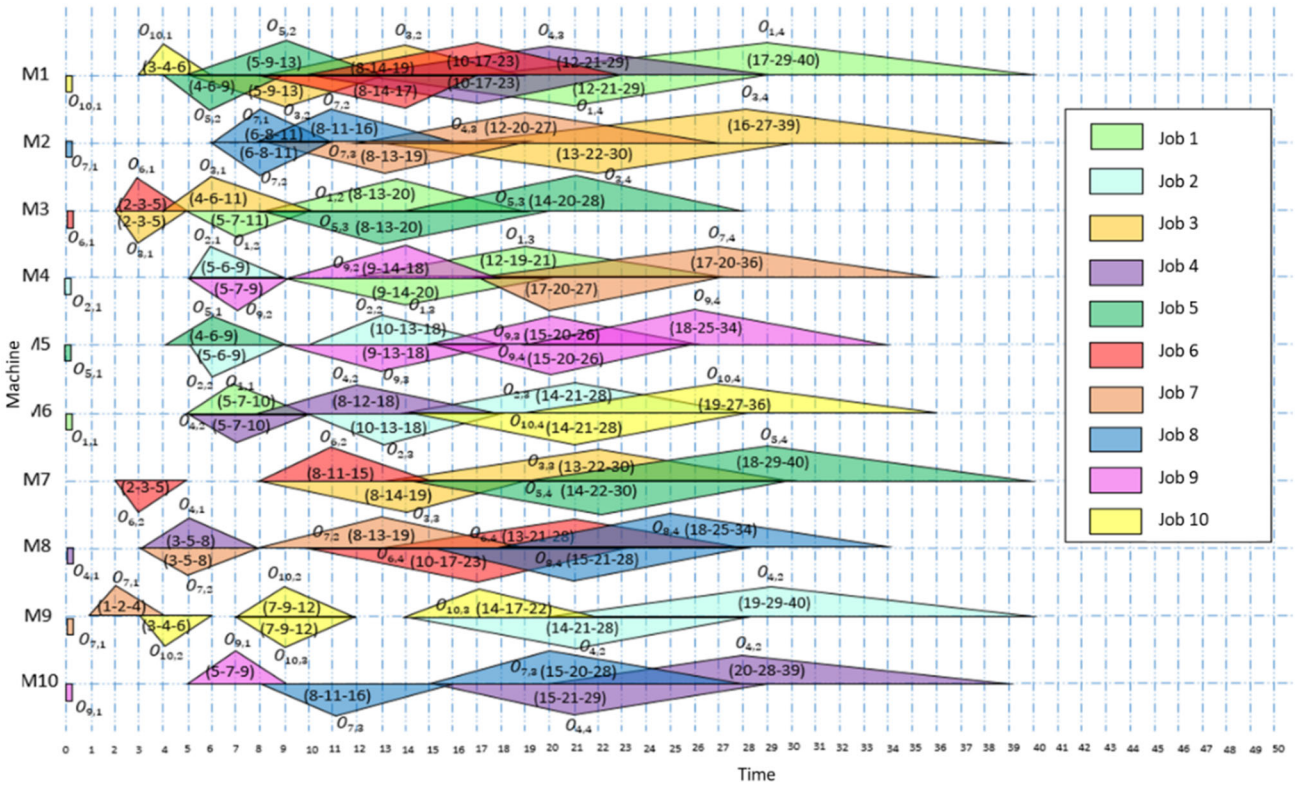


Fig. 8 The best solution for instance 1, obtained with the TLBO-A for the P-MOFfJSP

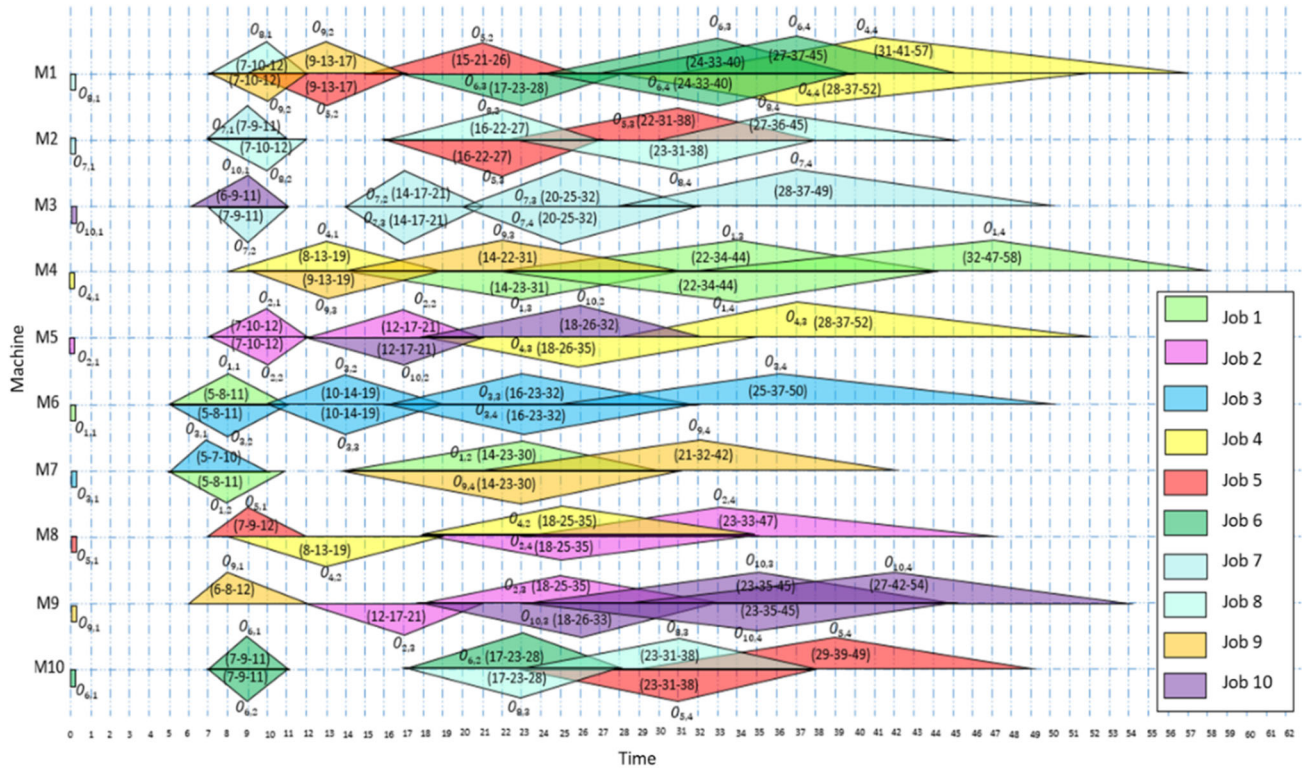


Fig. 9 The Best solution for instance 2, obtained with the TLBO-A for the P-MOFfJSP

M_5, M_6, M_8, and M_9 with four jobs and M_7 and M_{10}, with three jobs, respectively.

Figure 9 presents the best solution for instance 2, compared with the TLBO-A for the P-MOFfJSP, including the programming of the jobs for each one of the ten considered machines. In the case of job 1, its four operations must be developed on machines M_6, M_7, and M_4. Job 2 is scheduled on machines M_5, M_9, and M_8. Job 3 is scheduled on machines M_7 and M_6. According to the results presented in Fig. 9, the machine that develops the most jobs is M_1, with six jobs, followed by M_{10}, M_2, M_3, M_4, M_5, M_6, and M_9, with four jobs and M_7 and M_8, with three jobs, respectively.

The performance of TLBO-A in the FfJSP environment to minimize the fuzzy Cmax was evaluated by conducting 18 experimental runs for the scenario of Total flexibility with the instances by Lin (2002). The results obtained from these runs are presented in the following table (Table 7):

Table 7 Results and comparisons of the instances by Lin (2002) for the Fuzzy Makespan (FfJSP)

Instance	Algorithm	Middle value	Best value	Worst value
1	TLBO-A	(21–32–45)	(21–32–45)	(21–32–45)
	HMVO	(21–32–45)	(21–32–45)	(21–32–45)
	BAB		(21–32–45)	
2	TLBO-A	(19–26–35)	(19–26–35)	(19–26–35)
	HMVO	(18–26–35)	(18–26–35)	(18–26–35)
	BAB		(18–26–35)	
3	TLBO-A	(11–21–29)	(11–21–29)	(11–21–29)
	HMVO	(13–20–26)	(13–20–26)	(13–20–26)
	BAB		(13–20–26)	

The best solution found with the TLBO-A algorithm for the first instance by Lin (2002) is shown in Fig. 10. The optimal solution for the fuzzy makespan for this instance coincides with the best solution. The fuzzy makespan (21-32-45) is the completion time of the operation O_{52} assigned to machine M_1.

The adaptive TLBO-A algorithm shows stability in solving small instances with partial flexibility, as evidenced by the TLBO-A algorithm's evolution in Fig. 11, applied to the first instance by Lin (2002). The figure displays only three out of the 21 points corresponding to the solution space, indicating that the solution remains constant once found. The best solution for this problem is represented by the fuzzy triangular number (21-32-45), with a weight of 32.5 for the fuzzy makespan and a fuzzy average flow of 29.41.

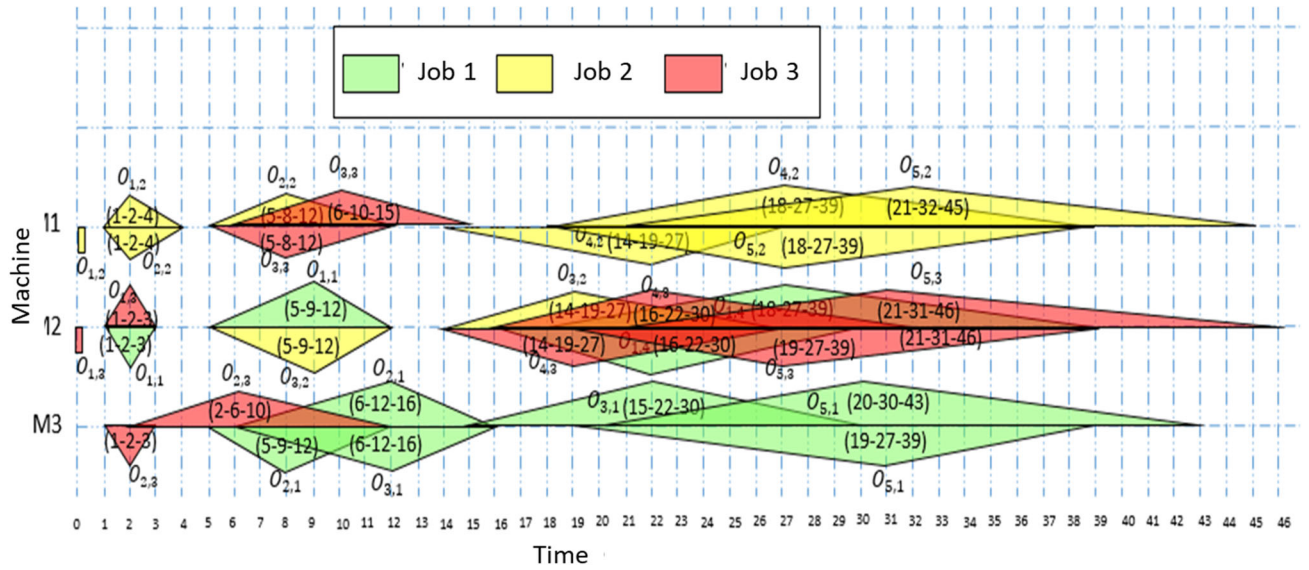


Fig. 10 The best solution of instance 1 (Lin (2002)) with the TLBO-A for MOFfJSP

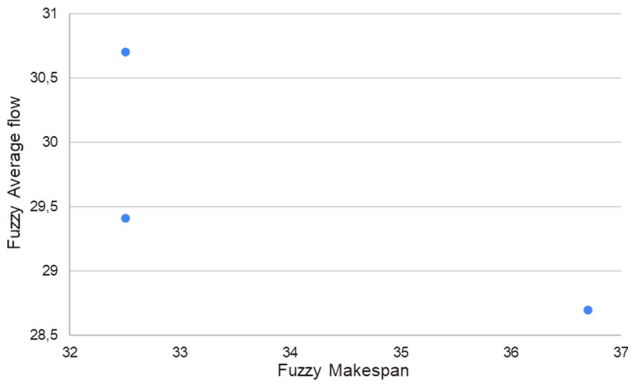


Fig. 11 Evolution of the TLBO-A in the best solution for instance 1 (Lin (2002))

5.1 Managerial implications

Manufacturing systems must establish strategies that generate agile responses to unexpected orders and product changes in less time. This can be achieved by using optimization algorithms for flexible Job-Shop problems. These algorithms can help manufacturers meet variations in demand and develop products with short cycle times while providing excellent physical and information integration of manufacturing system components.

The practical relevance of modeling the flexible manufacturing systems and the problem of production scheduling is a topic of great interest to researchers and manufacturing industries. By enhancing their production scheduling models, manufacturers can benefit from several advantages, including the following:

- Visualizing and executing strategies effectively in the production schedule
- Achieving customer loyalty through faster response times to orders and product changes.
- Reducing costs through optimized use of resources.
- Guaranteeing high-quality products through efficient scheduling and production processes.

Optimization algorithms for flexible Job-Shop problems provide a valuable tool for manufacturers to improve their production scheduling models and remain competitive in an ever-changing market.

6 Conclusions

The TLBO-A algorithm outperforms the standard TLBO algorithm on most instances of the FfJSP, except for instances 2 and 5, where it does not outperform the BSHH algorithm. However, when compared to the BSHH algorithm, it was found that the latter produces the best results in all instances for the fuzzy makespan problem. Despite this, the TLBO-A algorithm shows promising results compared to other existing algorithms, with a variation in the best results ranging from 2 to 11% across all instances. The proposed TLBO-A algorithm is a novel solution to optimize non-polynomial problems, such as the Multi-objective Flexible Job Shop Scheduling problem with fuzzy processing times. It incorporates genetic operators and an adaptive strategy for population reconfiguration, which enhances the search process for optimal solutions and complements the fuzzy processing time of the FfJSP, making it suitable for systems with partial or total flexibility within the work routes and available system resources. The TLBO-A is a hybrid improvement metaheuristic

that applies the mutation and crossover genetic operators in its exploration strategy to enhance and benefit from its advantages when solving non-polynomial problems such as the P-MOFfJSP. The proposed approach of the TLBO-A algorithm is integrative, where the sequence and assignment subproblems are addressed simultaneously. The multi-criteria optimization technique is used to solve the objective of the FfJSP and P-MOFfJSP, varying the weight percentage of the fuzzy makespan in a percentage range between zero (0) and one (1) to obtain the solutions or decisions of the problem.

In summary, the proposed TLBO-A algorithm presents a novel solution to optimize non-polynomial problems and shows promising results compared to other existing algorithms. It enhances the search process for optimal solutions and complements the fuzzy processing time of the FfJSP, making it suitable for systems with partial or total flexibility within the work routes and available system resources. Finally, one of the limitations of this study is the comparability with other multi-objective studies. It is important to note that the instances used in this study were developed by Lei (2010). To overcome this limitation, future studies could explore new and more challenging instances that better reflect real-world scenarios. This would provide a more comprehensive evaluation of the proposed TLBO-A algorithm's performance and its potential to solve real-world problems.

6.1 Future research

The proposed model for solving the P-MOFfJSP problem has several potential areas for future research. For example, include predictive aspects with machine learning for fuzzy production rescheduling scenarios (Adnan et al. 2018, 2022). The several directions to extending the model and evaluating its behavior with other optimization and hybridization techniques to overcome the limitations of the proposed model are outlines as follows:

Possible extensions:

- Evaluate the performance of TLBO-A with other multi-objective optimization methods and additional objectives for the P-MOFfJSP problem.
- Investigate the behavior of TLBO-A with other hybridization techniques and methods to improve its efficiency.
- Incorporate additional noise factors into the P-MOFfJSP to make it more representative of real-world scenarios, such as delivery and availability times.
- Study fuzzy environments by including fuzzy processing and delivery times to improve the model's practical applications.
- Use other crossing strategies for the different phases of the Teaching and Learning phase of the TLBO-A to enhance its results.
- Analyze the behavior of other criteria, such as a machine with less load, to generate assignments within the Adaptive phase.
- Incorporate preventive and corrective maintenance periods into the schedule to improve the model's practicality in real-world applications.
- Combine the TLBO Technique with other metaheuristics that may be different from the evolutionary ones, to identify potentialities in processing capacity.

Appendix

Appendix 1: TLBO-A: Main Class—Pseudocode.

```
Pseudo Main Class Code TLBO-A
Entry:Structure and Data of the FfJSP ;
Exit: Production Programming and convergence graph;
Process Graph_FfJSP_TLBO_ADAPTIVE_FINAL.Graph.Linear_TLBO {
    Execute Data_Entry_FfJSP_TLBO_ADAPTIVE_FINAL();
    x= 0;  // Solution index
    While      Beta <= 1        {
        ExecuteAlgorithm_FfJSP_TLBO_ADAPTIVE_FINAL() ;
        Accum_Obj_Cmax [x]= Cmax_Fuzzy ;
        Accum_Obj_Flow [x]= Flow_Average_Diffuse ;
        Beta = Beta+0.05;
        x=x+1 ;
    }end while
    Execute Graph_FfJSP_TLBO_ADAPTIVE_FINAL.Graph.Linear_TLBO () ;
} End Process
```

Appendix 2: TLBO-A: Pseudocode of the Data Entry Method.

```
Pseudo Code of the Data Entry Method
Entry:: Job_Number, Machine_Number, Job_Operation_Number, Fuzzy_Processing_Times,
Population_Size, Levels, Crossover_Probability, Mutation_Probability, %Best_Students, Mutation_Intensity;
Exit: FfJSP System Configuration ;
Process Data_Entry_FfJSP_TLBO_ADAPTIVE_FINAL {
    Read Excel file;
    Determine System Flexibility;
}End Process
```

Appendix 3: TLBO-A: Pseudocode for Initial Population Generation.

```
Pseudo Code of the Initial Population Generation Method
Entry:ProbAsig1, ProbAsig2, ProbAsig3
Exit:Initial Population
Process Generation_ Initial_ Population {
    Execute Data_Entry_FfJSP_TLBO_ADAPTIVE_FINAL ;
    While   p < Size_Ppopulation    {
        //Generation Vector Sequence of the initial population
        generate Initial_Sequence_Vector;// randomly
        generate Initial_Order_Vector;
        //GenerationAllocationVector of the initial population
        generate Random;
        ProbChoice = Random ;
        For  i ← 0  until  Total_Operations   {
            Yes  ProbChoice < ProbAsig1        {
                Assign_Vector[i] = Random_Asig;
            If not
                Yes  ProbAsig1 < ProbEleccion < ProbAsig2 ;     {
                    Asig_Vector[i] = Asig_SPT;
                If not
                    Yes  ProbAsig2 < ProbEleccion < ProbAsig3 ;     {
                        Asig_Vector[i] = Asig_LPT;
                    If not
                        Yes  ProbChoice > ProbAsig3    {
                            Vector_Asig[i] = Asig_SLM;
                        }End
                    }End
                }End
            }End
        }End
    }End
}End
```

Appendix 4: TLBO-A: Pseudocode for Mutation Operator Sequences.

```
Pseudo Operator Code Sequence Mutation
Entry:Cross population;
Exit: Mutated_Sequences [], Mutated_Order [], Mutated_Assignment;
Process Operator_ Mutation _of _sequences {
    Execute Generation_ Initial_ Population ;
    // Start Sequence mutation strategy
    Mutated_Students = 0, Population_to_mutate = 0, Intensity = 15, criteriachange = 60;
    Percentage_mutation= 0.4* level/Levels ;
    Population_to_mutate = Mutation_Percentage*Population_Size ;
    While    Mutated_Students < Population_to_mutate    {
        generate Random_Student ;
        Yes  MOD (count_BestStudent/changecriterion = 0 ) {
            WhileRandom_Student = 0                {
                generate Random_Student ;          //Random between 0 and Size_Population
            }end while
        } End
        While    Mutation_Intensity < Intensity    {
            generate Random_Oper_mutate ; // Random between 0 and Total_Operations
            Job_to_mutate = Sequence [Random_Oper_Mute][Random_Student];
            Order_Ope [][] = Order[Random_Oper_Mute][Random_Student];
            Saveorder data, sequence, job assignment to mutate;
            SaveJob operations positions to mutate;
            Mutatepositions of the operations of the mutated Work;
            Mutated_Sequence [];
            Mutated_Order [];
            Mutated_Assignment [];
            Mutation_Intensity++;
        }end while
increaseMutated_Students++;
}End Process
```

Appendix 5: TLBO-A Pseudocode for Mutation Operator of Assignments.

```
Pseudo Operator Code Assignment Mutation
Entry:Population with mutated sequences;
Exit: mutated assignments
ProcessOperator_ Mutation _of _Assignments {
    Execute Generation_ Initial_ Population ;
    // Start Machine Mutation - Less Time
    Mutated_Assignments1 = 0, Population_to_mutate = 0, Intensity = 15, Criteriachange = 60;
     Intensity_Mutation1;
    While     Allocations1_Mutated < Population_to_mutate  {
        generate Random1_Student ;
        Yes  MOD (count_BestStudent/changecriterion = 0 ) {
            WhileRandom1_Student = 0
                generate Random1_Student;      //Random between 0 and Size_Population
            }end while
        }End
        While      Intensity_Mutation1 < Intensity        {
            generate Random_Oper1_mutate ;     //Random between 0 and Total_Operations
            Assignment1_Muted [ Random_Oper1_mutar] = Machine_Timemin_Oper;
            Increase Intensity_Mutation1 ++ ;
         }end while
    Increase Assignments1_Muted ++;
    }end while
    // Start Machine Mutation - Random
    Mutated_Assignments2 = 0, Population_to_mutate = 0, Intensity = 15, Criteriachange = 60;
     Intensity_Mutation2;
    While     Allocations2_Mutated < Population_to_mutate  {
        generate Random2_Student ;
        Yes  MOD (count_BestStudent/changecriterion = 0 ) {
            WhileRandom2_Student = 0                {
                generate Random2_Student ;          // Random between 0 and Population_Size
            } end while
        }End yes
        While      Mutation_Intensity2 < Intensity        {
            generate Random_Oper2_mutate ;     // Random between 0 and Total_Operations
            Assignment2_Muted [ Random_Oper2_mutar]= Machine_Random_Oper;
        Increase Intensity_Mutation2 ++ ;
        } end while
    Increase Assignments2_Muted ++;
    } end while
    // Start Mutation of Machines - Operation in Machine of higher load to one of lower load
    Mutated_Assignments3 = 0, Population_to_mutate = 0, Intensity = 15, Criteriachange = 60;
     Intensity_Mutation3;
    While              Mutated_Assignments3 < Population_to_mutate{
        generate Random3_Student ;
        Yes  MOD (count_BestStudent/criterionchange = 0)  {
            WhileRandom3_Student = 0                {
                generate Random3_Student ;          // Random between 0 and Population_Size
            } end while
        }End
        While      Mutation_Intensity < Intensity   {
            generate Random_Oper_mutate ;      // Random between 0 and Total_Operations
             DetermineMore loaded machines;
             DetermineMachine less loaded;
            Yes   Assignment [Random_Oper3_mutate] = Maq_most_loaded []  {
                Assignment3_Muted [ Random_Oper3_mutate] = Maq_least_loaded;
            } End
        Increase Intensity_Mutation3 ++ ;
         }end while
    Increase Mutated_Assignments3 ++;
     }end while
}End Process
```

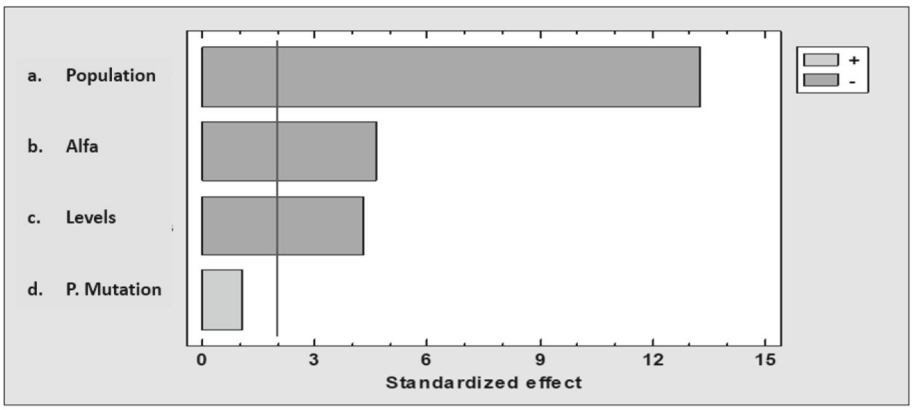
Appendix 6: TLBO-A: Pseudocode for Adaptive Strategy.

```
Adaptive Strategy Pseudo Code
Entry:Population with mutated assignments;
Exit: Reconfigured population;
Process Adaptive_Strategy {
    Yes   count_BestStudent > criteriachange   &   count_BestStudent < (4*changecriteria)   {
          %Poblacion_Reconfigure = 0.3*level/ Levels ;
          (1- %Best_Students) = 0.4 ;
     } End
    IFnot {
          %Poblacion_Reconfigurar = 0.7*level/ Levels ;
          (1- %Best_Students) = 0.6 ;
    } End
    Reconfigured_Students = %Reconfigured_Population*Population_Size;
    While      Reconfigured_Students < Population_Size          {
          generate sequence vector ;
          DetermineLoading of machines;
          DetermineOrder of operations;
          generate allocation vector ;
          Assignment [][] = Machine_less_load;
          DetermineLoading of machines;
          Increment Students_Reconfigured ++;
     } end
}End Process
```

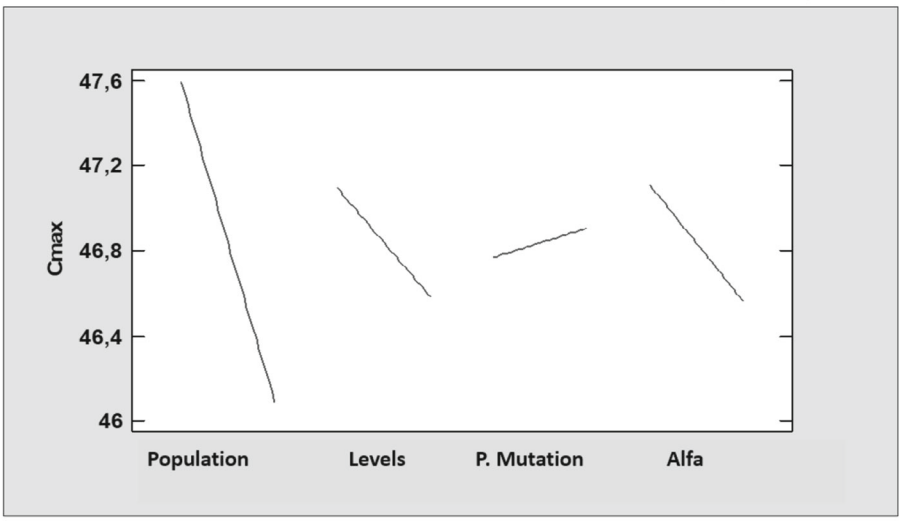
Appendix 7: TLBO-A: Pseudocode for the Convergence graph.

```
Pseudo Code Convergence Graph
Entry:Bestteacher [], Fuzzy_Cmax, Fuzzy_Average_Flow;
Exit: Convergence graph;
Process Graph_FfJSP_TLBO_ADAPTIVE_FINAL.Graph.Linear_TLBO {
     Execute Data_Entry_FfJSP_TLBO_ADAPTIVE_FINAL();
     Import data for charts ;
     For   i ← 0    until      x     {
          Cmax_Fuzzy = Pareto [i] ;
          Fuzzy_Average_Flow = Pareto2 [i] ;
          graphstring(Cmax_Fuzzy , Flow_Average_Fuzzy) ;
     }End
     To show On-screen graphics;
}End Process
```

Appendix 8: Result of the experimental design of the TLBO-A.

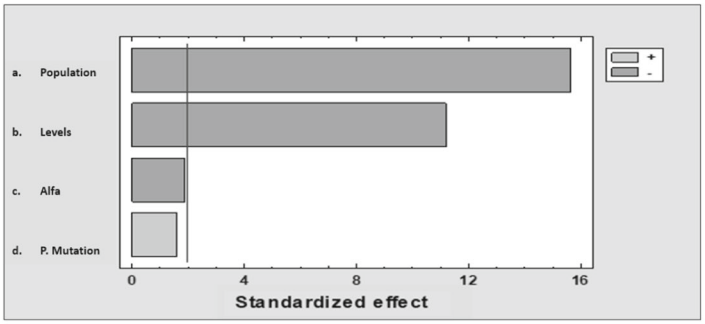


a Standardized Pareto chart for Cmax (FfJSP)

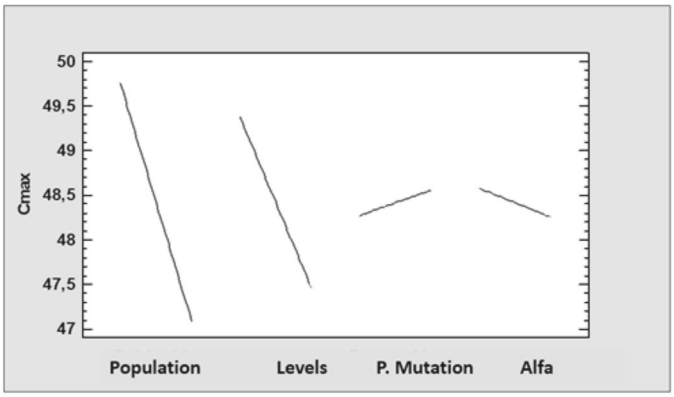


b Main Effects Plot for Cmax (FfJSP).

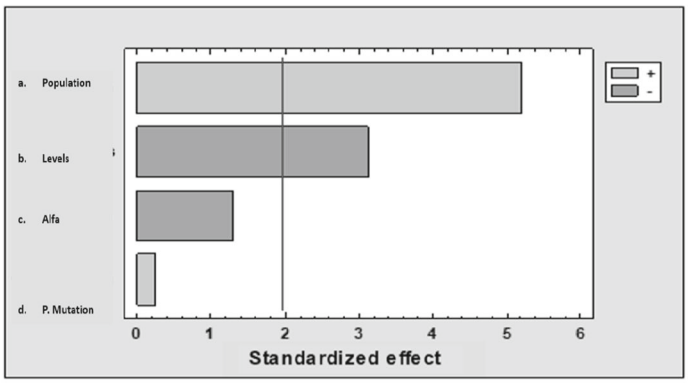
Appendix 9: Result of the experimental design of the P-MOFfJSP.



a. Standardized Pareto chart for the makespan.



b. Main Effects Plot for Cmax (P-MOFfJSP).



c. Standardized Pareto Chart for Flow (P-MOFfJSP).

d. Main Effects Diagram for Flow (P-MOFfJSP)

Author contributions MJT: Data curation, Methodology, Writing—Original draft preparation, Software. JA-C: Conceptualization, Methodology, Visualization, Validation. HO-M: Data curation, Methodology, Writing—Original draft preparation, Software. KS-N: Conceptualization, Methodology, Visualization, Validation. SSS: Supervision, Methodology, Writing—Reviewing and Editing.

Funding This work was completed at the author's own cost. Funding from a third party was not used to carry out this research work.

Data availability All data used to justify the proposed model are given in the manuscript.

Declarations

Conflicts of interest I do hereby declare that I do not have any conflicts of interest with other works. The study is not funded by any agency. The authors do hereby declare that there is no conflict of interest with other works regarding the publication of this paper.

Human and animal rights This article does not contain any studies with human participants or animals performed by any authors.

References

Abdullah S, Abdolrazzagh-Nezhad M (2014) Fuzzy job-shop scheduling problems: a review. Inf Sci 278:380–407

Acevedo Chedid J, Grice-Reyes J, Ospina-Mateus H, Salas-Navarro K, Santander-Mercado A, Sana SS (2020) Soft-computing approaches for rescheluding problems in a manufacturing industry. RAIRO Oper Res 55:S2125–S2159

Acevedo-Chedid J, Salas-Navarro K, Ospina-Mateus H, Villalobo A, Sana SS (2021) Production system in a collaborative supply chain considering deterioration. Int J Appl Comput Math 7:1–46

Adnan RM, Yuan X, Kisi O, Adnan M, Mehmood A (2018) Stream flow forecasting of poorly gauged mountainous watershed by least square support vector machine, fuzzy genetic algorithm and M5 model tree using climatic data from nearby station. Water Resour Manag 32:4469–4486

Adnan RM, Mostafa RR, Elbeltagi A, Yaseen ZM, Shahid S, Kisi O (2022) Development of new machine learning model for streamflow prediction: case studies in Pakistan. Stoch Env Res Risk Assess 36:999–1033

Al-Janabi S, Alkaim A (2020) A nifty collaborative analysis to predicting a novel tool (DRFLLS) for missing values estimation. Soft Comput 24:555–569

Al-Janabi S, Alkaim A (2022) A novel optimization algorithm (Lion-AYAD) to find optimal DNA protein synthesis. Egypt Inform J 23:271–290

Al-Janabi S, Alkaim A, Adel Z (2020a) An innovative synthesis of deep learning techniques (DCapsNet & DCOM) for generation electrical renewable energy from wind energy. Soft Comput 24:10943–10962

Al-Janabi S, Mohammad M, Al-Sultan A (2020b) A new method for prediction of air pollution based on intelligent computation. Soft Comput 24:661–680

Al-Janabi S, Alkaim A, Al-Janabi E, Alieboree A, Mustaja M (2021) Intelligent forecaster of concentrations (PM2.5, PM10, NO_2, CO, O_3, SO_2) caused air pollution (IFCsAP). Neural Comput Appl 33:14199–14229

Basiri MA, Alinezhad E, Tavakkoli-Moghaddam R, Shahsavari-Poure N (2020) A hybrid intelligent algorithm for a fuzzy multi-objective job shop scheduling problem with reentrant workflows and parallel machines. J Intell Fuzzy Syst 39:7769–7785

Baykasoğlu A, Hamzadayi A, Köse SY (2014) Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: flow shop and job shop scheduling cases. Inf Sci 276:204–218

Behnamian J (2017) Matheuristic for the decentralized factories scheduling problem. Appl Math Model 47:668–684

Boyer V, Vallikavungal J, Rodríguez XC, Salazar-Aguilar MA (2021) The generalized flexible job shop scheduling problem. Comput Ind Eng 160:107542

Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41:157–183

Braune R, Benda F, Doerner KF, Hartl RF (2022) A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. Int J Prod Econ 243:108342

Bulbul SMA, Roy PK (2014) Adaptive teaching learning based optimization applied to nonlinear economic load dispatch problem. Int J Swarm Intell Res 5:1–16

Chen JC, Wu CC, Chen CW, Chen KH (2012) Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm. Expert Syst Appl 39:10016–10021

Chiandussi G, Codegone M, Ferrero S, Varesio FE (2012) Comparison of multi-objective optimization methodologies for engineering applications. Comput Math Appl 63:912–942

Chiang TC, Lin HJ (2013) A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. Int J Prod Econ 141:87–98

Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219:8121–8144

Deng Q, Gong G, Gong X, Zhang L, Liu W, Ren Q (2017) A bee evolutionary guiding nondominated sorting genetic algorithm II for multiobjective flexible job-shop scheduling. Comput Intell Neurosci 2017:5232518

Engin O, Yilmaz MK, Baysal M, Sarucan A (2013) Solving fuzzy job shop scheduling problems with availability constraints using a scatter search method. J Mult Valued Log Soft Comput 21:317–334

Ertenlice O, Kalayci CB (2018) A survey of swarm intelligence for portfolio optimization: algorithms and applications. Swarm Evol Comput 39:36–52

Fazel Zarandi MH, Sadat Asl AA, Sotudian S, Castillo O (2020) A state of the art review of intelligent scheduling. Artif Intell Rev 53:501–593

Gaham M, Bouzouia B, Achour N (2018) An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion. Appl Intell 48:1423–1441

Gao J, Gen M, Sun L, Zhao X (2007) A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Comput Ind Eng 53:149–162

Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35:2892–2907

Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF (2015) An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time. Int J Prod Res 53:5896–5911

Gao KZ, Suganthan PN, Pan QK, Chua TJ, Chong CS, Cai TX (2016a) An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. Expert Syst Appl 65:52–67

Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF, Sadollah A (2016b) Artificial bee colony algorithm for scheduling and rescheduling

fuzzy flexible job shop problem with new job insertion. Knowl Based Syst 109:1–16
Gao D, Wang GG, Pedrycz W (2020) Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. IEEE Trans Fuzzy Syst 28:3265–3275
He C, Qiu D, Guo H (2013) Solving fuzzy job shop scheduling problem based on interval number theory. In: Proceedings of the 2012 international conference on information technology and software engineering. Springer, Berlin, Heidelberg
Ji X, Ye H, Zhou J, Yin Y, Shen X (2017) An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry. Appl Soft Comput 57:504–516
Jia S, Hu ZH (2014) Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem. Comput Oper Res 47:11–26
Jin L, Zhang C, Shao X, Tian G (2016) Mathematical modeling and a memetic algorithm for the integration of process planning and scheduling considering uncertain processing times. Proc Inst Mech Eng Part B J Eng Manuf 230:1272–1283
Jin L, Zhang C, Wen X, Sun C, Fei X (2021) A neutrosophic set-based TLBO algorithm for the flexible job-shop scheduling problem with routing flexibility and uncertain processing times. Complex Intell Syst 7:2833–2853
Joo BJ, Shim SO, Chua TJ, Cai TX (2018) Multi-level job scheduling under processing time uncertainty. Comput Ind Eng 120:480–487
Kacem I, Hammadi S, Borne P (2002) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60:245–276
Kaplanoğlu V (2016) An object-oriented approach for multi-objective flexible job-shop scheduling problem. Expert Syst Appl 45:71–84
Kato ERR, de Aguiar Aranha GD, Tsunaki RH (2018) A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and random-restart hill climbing. Comput Ind Eng 125:178–189
Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80:8091–8126
Kisi O, Parmar KS, Mahdavi-Meymand A, Adnan RM, Shahid S, Zounemat-Kermani M (2023) Water quality prediction of the yamuna river in India using hybrid neuro-fuzzy models. Water 15:1095
Kundakcı N, Kulak O (2016) Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. Comput Ind Eng 96:31–51
Lei D (2010) A genetic algorithm for flexible job shop scheduling with fuzzy processing time. Int J Prod Res 48:2995–3013
Lei D (2012) Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling. Appl Soft Comput 12:2237–2245
Lei H, Xing K, Han L, Gao Z (2017) Hybrid heuristic search approach for deadlock-free scheduling of flexible manufacturing systems using Petri nets. Appl Soft Comput 55:413–423
Lei K, Guo P, Zhao W, Wang Y, Qian L, Meng X, Tang L (2022) A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. Expert Syst Appl 205:117796
Li X, Gao L (2016) An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 174:93–110
Li JQ, Pan QK (2013a) Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities. Int J Prod Econ 145:4–17
Li JQ, Pan YX (2013b) A hybrid discrete particle swarm optimization algorithm for solving fuzzy job shop scheduling problem. Int J Adv Manuf Technol 66:583–596
Li JQ, Pan QK, Liang YC (2010) An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 59:647–662
Li X, Peng Z, Du B, Guo J, Xu W, Zhuang K (2017) Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems. Comput Ind Eng 113:10–26
Li JQ, Liu ZM, Li C, Zheng ZX (2021) Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem. IEEE Trans Fuzzy Syst 29:3234–3248
Li R, Gong W, Wang L, Lu C, Jiang S (2022a) Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time. Swarm Evol Comput 74:101139
Li R, Gong W, Lu C (2022b) A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling. Expert Syst Appl 203:117380
Li R, Gong W, Lu C (2022c) Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time. Comput Ind Eng 168:108099
Li J, Pan QK, Suganthan PN, Tasgetiren MF (2012) Solving fuzzy jJob-shop scheduling problem by a hybrid PSO algorithm. Swarm and Evolutionary Computation, Berlin, Heidelberg, Springer Berlin Heidelberg. Lecture Notes in Computer Science book series (LNTCS,vol 7269), pp 275–282
Lin FT (2002) Fuzzy job-shop scheduling based on ranking level (/spl lambda/, 1) interval-valued fuzzy numbers. IEEE Trans Fuzzy Syst 10:510–522
Lin J (2019) Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time. Eng Appl Artif Intell 77:186–196
Lin J, Zhang S (2016) An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. Comput Ind Eng 97:128–136
Lin J, Zhu L, Wang ZJ (2019) A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem. Comput Ind Eng 127:1089–1100
Liu B, Fan Y, Liu Y (2015) A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. Comput Ind Eng 87:193–201
Mandal B, Roy PK (2013) Optimal reactive power dispatch using quasi-oppositional teaching learning based optimization. Int J Electr Power Energy Syst 53:123–134
Mane SU, Adamuthe AC, Omane RR (2020) Master-Slave TLBO algorithm for constrained global optimization problems. EAI Endorsed Trans Scalable Inf Syst 8:e2
Mastrolilli M, Gambardella LM (2000) Effective neighbourhood functions for the flexible job shop problem. J Sched 3:3–20
Mohammed GS, Al-Janabi S (2022) An innovative synthesis of optmization techniques (FDIRE-GSK) for generation electrical renewable energy from natural resources. Results Eng 16:100637. https://doi.org/10.1016/j.rineng.2022.100637
Ortíz-Barrios M, Petrillo A, De Felice F, Jaramillo-Rueda N, Jiménez-Delgado G, Borrero-López L (2021) A dispatching-fuzzy AHP-TOPSIS model for scheduling flexible job-shop systems in industry 4.0 context. Appl Sci 11:5107
Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34:1539–1548
Palacios JJ, Puente J, González-Rodríguez I, Vela CR (2013) Hybrid tabu search for fuzzy job shop. Natural and artificial models in computation and biology. Springer, Berlin, Heidelberg

Palacios JJ, González MA, Vela CR, González-Rodríguez I, Puente J (2015) Genetic tabu search for the fuzzy flexible job shop problem. Comput Oper Res 54:74–89

Pan C, Qiao Y, Wu N, Zhou M (2014) A novel algorithm for wafer sojourn time analysis of single-arm cluster tools with wafer residency time constraints and activity time variation. IEEE Trans Syst Man Cybern Syst 45:805–818

Pan Z, Lei D, Wang L (2022) A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling. IEEE Trans Syst Man Cybern Syst 52:5295–5307

Petrović DV, Tanasijević M, Milić V, Lilić N, Stojadinović S, Svrkota I (2014) Risk assessment model of mining equipment failure based on fuzzy logic. Expert Syst Appl 41:8157–8164

Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35:3202–3212

Pickard JK, Carretero JA, Bhavsar VC (2016) On the convergence and origin bias of the teaching-learning-based-optimization algorithm. Appl Soft Comput 46:115–127

Pinedo M (2005) Planning and scheduling in manufacturing and services. Springer

Rao RV, Rai DP (2016) Optimisation of advanced finishing processes using a teaching-learning-based optimisation algorithm. Nanofinishing science and technology. CRC Press, pp 495–518

Roy PK, Sarkar R (2014) Solution of unit commitment problem using quasi-oppositional teaching learning based algorithm. Int J Electr Power Energy Syst 60:96–106

Satapathy S, Naik A, Parvathi K (2013) A teaching learning based optimization based on orthogonal design for solving global optimization problems. Springerplus 2:1–12

Seck-Tuoh-Mora JC, Escamilla-Serna NJ, Montiel-Arrieta LJ, Barragan-Vite I, Medina-Marin J (2022) A global neighborhood with hill-climbing algorithm for fuzzy flexible job shop scheduling problem. Mathematics 10:4233

Seyyedi MH, Saghih AMF, Azimi ZN (2021) A fuzzy mathematical model for multi-objective flexible job-shop scheduling problem with new job insertion and earliness/tardiness penalty. Int J Ind Eng Theory Appl Pract, 28

Shao W, Pi D, Shao Z (2017) An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem. Appl Soft Comput 61:193–210

Shi D, Zhang B, Li Y (2020) A multi-objective flexible job-shop scheduling model based on fuzzy theory and immune genetic algorithm. Int J Simul Model 19:123–133

Song H, Liu P (2022) A study on the optimal flexible job-shop scheduling with sequence-dependent setup time based on a hybrid algorithm of improved quantum cat swarm optimization. Sustainability 14:9547

Sun L, Lin L, Gen M, Li H (2019) A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling. IEEE Trans Fuzzy Syst 27:1008–1022

Thammano A, Teekeng W (2015) A modified genetic algorithm with fuzzy roulette wheel selection for job-shop scheduling problems. Int J Gen Syst 44:499–518

Tran TD, Varela R, González-Rodríguez I, Talbi EG (2014) Solving fuzzy job-shop scheduling problems with a multiobjective optimizer. *Knowledge and systems engineering*. Springer International Publishing, Cham

Wang L, Wang S, Xu Y, Zhou G, Liu M (2012a) A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. Comput Ind Eng 62:917–926

Wang X, Gao L, Zhang C, Li X (2012b) A multi–objective genetic algorithm for fuzzy flexible job–shop scheduling problem. Int J Comput Appl Technol 45:115–125

Wang X, Li W, Zhang Y (2013) An improved multi–objective genetic algorithm for fuzzy flexible job–shop scheduling problem. Int J Comput Appl Technol 47:280–288

Wang S, Liu G, Gao S (2016) A hybrid discrete imperialist competition algorithm for fuzzy job-shop scheduling problems. IEEE Access 4:9320–9331

Wang C, Tian N, Ji Z, Wang Y (2017) Multi-objective fuzzy flexible job shop scheduling using memetic algorithm. J Stat Comput Simul 87:2828–2846

Wang GG, Gao D, Pedrycz W (2022) Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. IEEE Trans Industr Inf 18:8519–8528

Xia W, Wu Z (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput Industr Eng 48(2):409–425. https://doi.org/10.1016/j.cie.2005.01.018

Xing LN, Chen YW, Yang KW (2009) An efficient search method for multi-objective flexible job shop scheduling problems. J Intell Manuf 20:283–293

Xu Y, Wang L, Wang SY, Liu M (2015) An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. Neurocomputing 148:260–268

Xu Y, Peng Y, Su X, Yang Z, Ding C, Yang X (2022) Improving teaching–learning-based-optimization algorithm by a distance-fitness learning strategy. Knowl Based Syst 257:108271

Xu L, Xia ZY, Ming H (2016) Study on improving multi-objective flexible job shop scheduling based on memetic algorithm in the NSGA-II framework. In: 2016 2nd international conference on cloud computing and internet of things (CCIOT), IEEE. Conference Location: Dalian, China

Yu D, Hong J, Zhang J, Niu Q (2018) Multi-objective individualized-instruction teaching-learning-based optimization algorithm. Appl Soft Comput 62:288–314

Zadeh L (1963) Optimality and non-scalar-valued performance criteria. IEEE Trans Autom Control 8:59–60

Zhang G, Gao L, Shi Y (2011) An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Syst Appl 38:3563–3573

Zhang G, Lu X, Liu X, Zhang L, Wei S, Zhang W (2022) An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown. Expert Syst Appl 203:117460

Zheng YL, Li YX (2012) Artificial bee colony algorithm for fuzzy job shop scheduling. Int J Comput Appl Technol 44:124–129

Zheng YL, Li YX, Lei DM (2012) Multi-objective swarm-based neighborhood search for fuzzy flexible job shop scheduling. Int J Adv Manuf Technol 60:1063–1069

Zou F, Chen D, Xu Q (2019) A survey of teaching–learning-based optimization. Neurocomputing 335:366–383

Authors and Affiliations

Mary Jiménez Tovar[1] · Jaime Acevedo-Chedid[1] · Holman Ospina-Mateus[1] · Katherinne Salas-Navarro[2] · Shib Sankar Sana[3]

✉ Shib Sankar Sana
shib_sankar@yahoo.com

Mary Jiménez Tovar
maryjimenez@utb.edu.co

Jaime Acevedo-Chedid
jacevedo@utb.edu.co

Holman Ospina-Mateus
hospina@utb.edu.co

Katherinne Salas-Navarro
ksalas2@cuc.edu.co

[1] Department of Industrial Engineering, Universidad Tecnológica de Bolívar, Cartagena, Colombia

[2] Department of Productivity and Innovation, Universidad de la Costa, Barranquilla, Colombia

[3] Department of Mathematics, Kishore Bharati Bhagini Nivedita College, Behala, Kolkata 700060, India