Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.049481

ARTICLE

An Elite-Class Teaching-Learning-Based Optimization for Reentrant Hybrid Flow Shop Scheduling with Bottleneck Stage

Deming Lei, Surui Duan, Mingbo Li* and Jing Wang

College of Automation, Wuhan University of Technology, Wuhan, 430070, China

*Corresponding Author: Mingbo Li. Email: feng10111217@whut.edu.cn

Received: 09 January 2024 Accepted: 11 March 2024 Published: 25 April 2024

ABSTRACT

Bottleneck stage and reentrance often exist in real-life manufacturing processes; however, the previous research rarely addresses these two processing conditions in a scheduling problem. In this study, a reentrant hybrid flow shop scheduling problem (RHFSP) with a bottleneck stage is considered, and an elite-class teaching-learning-based optimization (ETLBO) algorithm is proposed to minimize maximum completion time. To produce high-quality solutions, teachers are divided into formal ones and substitute ones, and multiple classes are formed. The teacher phase is composed of teacher competition and teacher teaching. The learner phase is replaced with a reinforcement search of the elite class. Adaptive adjustment on teachers and classes is established based on class quality, which is determined by the number of elite solutions in class. Numerous experimental results demonstrate the effectiveness of new strategies, and ETLBO has a significant advantage in solving the considered RHFSP.

KEYWORDS

Hybrid flow shop scheduling; reentrant; bottleneck stage; teaching-learning-based optimization

1 Introduction

A hybrid flow shop scheduling problem (HFSP) is a typical scheduling problem that exists widely in many industries such as petrochemicals, chemical engineering, and semiconductor manufacturing [1,2]. The term 'reentrant' means a job may be processed multiple times on the same machine or stage [3]. A typical reentrant is a cyclic reentrant [4,5], which means that each job is cycled through the manufacturing process. As an extension of HFSP, RHFSP is extensively used in electronic manufacturing industries, including printed circuit board production [6] and semiconductor wafer manufacturing [7], etc.

RHFSP has been fully investigated and many results have been obtained in the past decade. Xu et al. [8] applied an improved moth-flame optimization algorithm to minimize maximum completion time and reduce the comprehensive impact of resources and environment. Zhou et al. [9] proposed a hybrid differential evolution algorithm with an estimation of distribution algorithm to minimize total weighted completion time. Cho et al. [10] employed a Pareto genetic algorithm with a local search strategy and Minkowski distance-based crossover operator to minimize maximum completion time and total tardiness. Shen et al. [11] designed a modified teaching-learning-based optimization (TLBO) algorithm to minimize maximum completion time and total tardiness, where Pareto-based ranking method and training phase are adopted.

In recent years, RHFSP with real-life constraints has attracted much attention. Lin et al. [12] proposed a hybrid harmony search and genetic algorithm (HHSGA) for RHFSP with limited buffer to minimize weighted values of maximum completion time and mean flowtime. For RHFSP with missing operations, Tang et al. [13] designed an improved dual-population genetic algorithm (IDPGA) to minimize maximum completion time and energy consumption. Zhang et al. [14] considered machine eligibility constraints and applied a discrete differential evolution algorithm (DDE) with a modified crossover operator to minimize total tardiness. Chamnanlor et al. [15] adopted a genetic algorithm hybridized ant colony optimization for the problem with time window constraints. Wu et al. [16] applied an improved multi-objective evolutionary algorithm based on decomposition to solve the problem with bottleneck stage and batch processing machines.

In HFSP with H stages, each job is processed in the following sequence: Stage 1, stage 2, $\cdots$, stage H. If processing time of each job at a stage is significantly longer than its processing time at other stages, then that stage is the bottleneck stage. The bottleneck stages often occur in real-life manufacturing processes when certain stages of the process are slower than others, limiting the overall efficiency of the process [16–21]. These stages may arise due to resource constraints, process complexity or other factors. Bottleneck stage is a common occurrence in real-life manufacturing processes, such as seamless steel tube cold drawing production [16], engine hot-test production [20] and casting process [21]. More processing resources or times are needed at the bottleneck stage, and the production capacity of the whole shop will be limited because of bottleneck stage. There are some works about HFSP with the bottleneck stage. Costa et al. [17] considered HFSP with bottleneck stage and limited human resource constraint and applied a novel discrete backtracking search algorithm. Shao et al. [18] designed an iterated local search algorithm for HFSP with the bottleneck stage and lot-streaming. Liao et al. [19] developed a new approach hybridizing particle swarm optimization with bottleneck heuristic to fully exploit the bottleneck stage in HFSP. Zhang et al. [20] studied a HFSP with limited buffers and a bottleneck stage on the second process routes and proposed a discrete whale swarm algorithm to minimize maximum completion time. Wang et al. [21] adopted an adaptive artificial bee colony algorithm for HFSP with batch processing machines and bottleneck stage.

As stated above, RHFSP with real-life constraints such as machine eligibility and limited buffer has been investigated; however, RHFSP with bottleneck stage is seldom considered, which exists in real-life manufacturing processes such as seamless steel tube cold drawing production [16]. The modelling and optimization on reentrance and bottleneck stage can lead to optimization results with high application value, so it is necessary to deal with RHFSP with the bottleneck stage.

TLBO [22–26] is a population-based algorithm inspired by passing on knowledge within a classroom environment and consists of the teacher phase and learner phase. TLBO [27–31] has become a main approach to production scheduling [32–35] due to its simple structure and fewer parameters. TLBO has been successfully applied to solve RHFSP [11] and its searchability and advantages on RHFSP are tested; however, it is rarely used to solve RHFSP with the bottleneck stage, which is an extension of RHFSP. The successful applications of TLBO to RHFSP show that TLBO has potential advantages to address RHFSP with bottleneck stage, so TLBO is chosen.

In this study, the reentrance and bottleneck stages are simultaneously investigated in a hybrid flow shop, and an elite-class teaching-learning-based optimization (ETLBO) is developed. The main contributions can be summarized as follows: (1) RHFSP with bottleneck stage is solved and a new algorithm called ETLBO is proposed to minimize maximum completion time. (2) In ETLBO, teachers are divided into formal ones and substitute ones. The teacher phase consists of teacher competition and teacher teaching, the learner phase is replaced by reinforcement research of elite class; adaptive adjustment on teachers and classes is applied based on class quality, and class quality is determined by the number of elite solutions in class. (3) Extensive experiments are conducted to test the performances of ETLBO by comparing it with other existing algorithms from the literature. The computational results demonstrate that new strategies are effective and ETLBO has promising advantages in solving RHFSP with bottleneck stage.

The remainder of the paper is organized as follows. The problem description is described in Section 2. Section 3 shows the proposed ETLBO for RHFSP with the bottleneck stage. Numerical test experiments on ETLBO are reported in Section 4. Conclusions and some topics of future research are given in the final section.

2 Problem Description

RHFSP with bottleneck stage is described as follows. There are n jobs $J_1, J_2, \cdots, J_n$ and a hybrid flow shop with H stages. Stage k has $S_k \geq 1$ machines $M_{k1}, M_{k2}, \cdots, M_{kS_k}$, and at least one stage exists two or more identical parallel machines. Each job is processed $L\,(L > 1)$ times in the following sequence: Stage 1, stage 2, $\cdots$, stage H, which means each job is reentered $L-1$ times. Each job must be processed in the last H stages before next processing can begin until its L processings are finished. p_{ik} represents the processing time of job J_i at stage k.There is a bottleneck stage b, $b \in (1, H)$. p_{ib} is often more than about $10 \times p_{ik}$ such as casting process [21], $k \neq b$.

There are the following constraints on jobs and machines:

All jobs and machines are available at time 0.

Each machine can process at most one operation at a time.

No jobs may be processed on more than one machine at a time.

Operations cannot be interrupted.

The problem can be divided into two sub-problems: scheduling and machine assignment. Scheduling is applied to determine processing sequence for all jobs on each machine. Machine assignment is used for selecting appropriate machine at each stage for each job. There are strong coupled relationships between these two sub-problems. The optimization contents of scheduling are directly determined by the machine assignment. To obtain an optimal solution, it is necessary to efficiently combine the two sub-problems.

The goal of the problem is to minimize maximum completion time when all constraints are met.

$$C_{\max} = \max_{i=1,2,\cdots,n} \{C_i\} \tag{1}$$

where C_i is the completion time of job J_i, and $C_{\max}$ denotes maximum completion time.

An example is shown in Table 1, where $n = 5$, $H = 3$, $L = 2$, $b = 2$, $S_1 = 2$, $S_2 = 4$, $S_3 = 3$. A schedule of the example with $C_{\max} = 749$ is displayed in Fig. 1. O_{ik}^{l} denotes the operation in which job J_i is processed for the *l-th* time at stage k.

Table 1: An example of RHFSP

Job (i)	p_{i1}	p_{i2}	p_{i3}
1	12	237	18
2	19	290	11
3	16	278	12
4	12	221	12
5	16	261	13

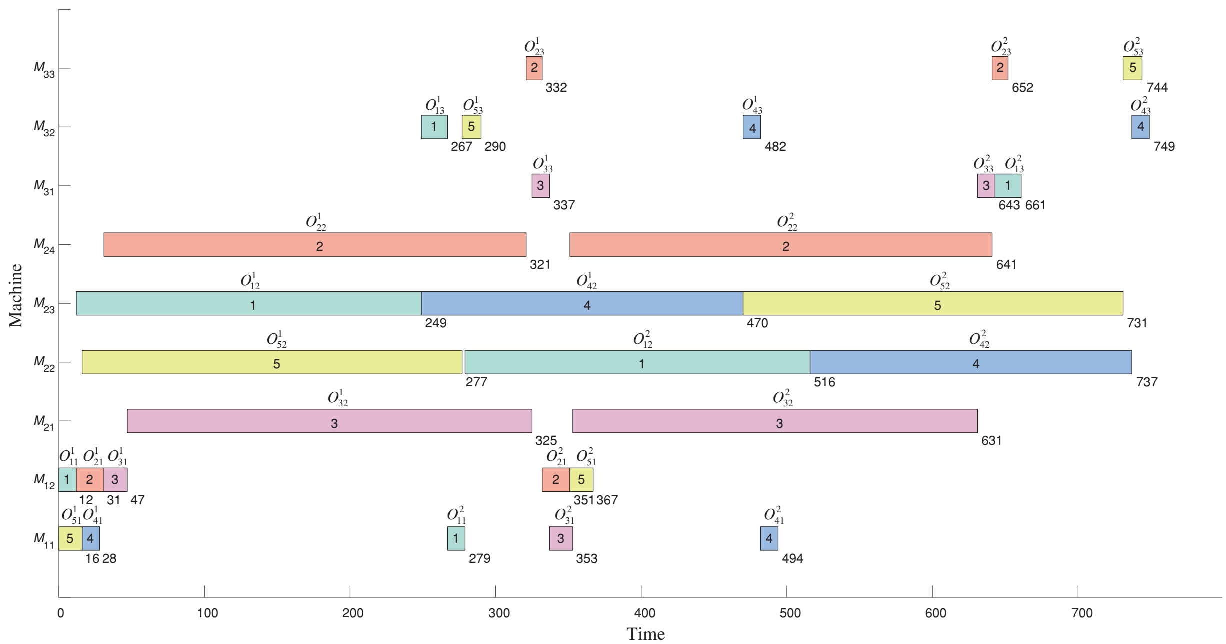

Figure 1: A schedule of the example

3 ETLBO for RHFSP with Bottleneck Stage

Some works are obtained on TLBO with multiple classes; however, in the existing TLBO [36–39], competition among teachers is not used, reinforcement search of some elite solutions and adaptive adjustment on classes and teachers are rarely considered. To effectively solve RHFSP with bottleneck stage, ETLBO is constructed based on reinforcement search of elite class and adaptive adjustment.

3.1 Initialization and Formation of Multiple Classes

To solve the considered RHFSP with reentrant feature, a two-string representation is used [12]. For RHFSP with n jobs, H stages and L processing, its solution is represented by a machine assignment string $[q_{11}, q_{12}, \cdots, q_{1H\times L}|q_{21}, q_{22}, \cdots, q_{2H\times L}|\cdots|q_{n1}, q_{n2}, \cdots, q_{nH\times L}]$ and a scheduling string $[\pi_1, \pi_2, \cdots, \pi_{n\times H\times L}]$, where $\pi_i \in [1, 2, \cdots, n]$, $q_{i((l-1)\times H+k)}$ is the machine for the *l-th* processing at stage k for job J_i.

In scheduling string, the frequency of occurrence is $H \times L$ for each job J_i. Take job J_1 as an example, when $g < H$, the g-th 1 corresponds to O^1_{1g}; when $H < g \le 2H$, the g-th 1 denotes O^2_{1g}, and so on. The whole machine assignment string is divided into n segments, each segment corresponds to the assigned machines at all stages in the l-th processing for a job.

The decoding procedure to deal with reentrant feature is shown below. Start with job π_1, for each job π_i, decide its corresponding operation $O^l_{\pi_i g}$, which is processed on a assigned machine for $O^l_{\pi_i g}$ by machine assignment string.

For the example in Section 2, the solution is shown in Fig. 2. For job J_4, a segment of [1, 3, 2, 1, 2, 2] is obtained from machine assignment string, in the segment, 1, 3, 2 means that operation O^1_{41}, O^1_{42}, O^1_{43} are processed on machines M_{11}, M_{23}, M_{32} respectively in the first processing, completion times of three operations are 28, 470, 482; 1, 2, 2 indicates that O^2_{41}, O^2_{42}, O^2_{43} are processed on machines M_{11}, M_{22}, M_{32} in the second processing, their corresponding completion times are 494, 737, 749, respectively. A schedule of the decoding as shown in Fig. 1.

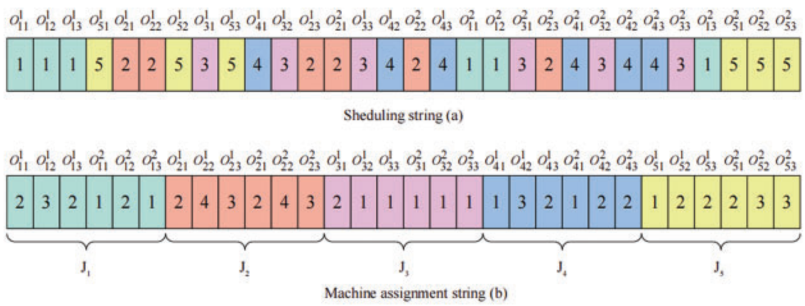

Figure 2: A coding of the example

Initial population P with N solutions are randomly produced.

The formation of multiple classes is described as follows:

1. Sort all solutions of P in ascending order of $C_{\max}$, suppose that $C_{\max}(x_1) \le C_{\max}(x_2) \le \cdots \le C_{\max}(x_N)$, first $(\alpha + \beta)$ solutions are chosen as teachers and formed as a set Ω, and remaining solutions are learners.
2. Divide all learners into α classes by assigning each learner x_i to class $Cls_{(i-1)(\mathrm{mod}\ \alpha)+1}$.
3. Each class Cls_r is assigned a formal teacher in the following way, $r = 1$, repeat the following steps until $r > \alpha$: Randomly select a teacher from Ω as the formal teacher $x^r_{teacher}$ of Cls_r, $\Omega = \Omega \backslash x^r_{teacher}$, $r = r + 1$.

where $C_{\max}(x)$ denotes the maximum completion time of solution x.

The remaining β solutions in Ω are regarded as substitute teachers, $\Omega = \left\{x^{\alpha+1}_{teacher}, \cdots, x^{\alpha+\beta}_{teacher}\right\}$. Teachers are not assigned to classes, and each class consists only of learners.

3.2 Search Operators

Global search $GS(x, y)$ is described as follows. If $rand \le 0.5$, then order-based crossover [12] is done on scheduling string of x and y; otherwise, two-point crossover [40] is executed on machine assignment string of x and y, a new solution z is obtained, if $C_{\max}(z) < C_{\max}(x)$, then replace x with z, where random number $rand$ follows uniform distribution on [0, 1].

Ten neighborhood structures $\mathcal{N}_1 - \mathcal{N}_{10}$ are designed, $\mathcal{N}_1 - \mathcal{N}_5$ are about scheduling string and $\mathcal{N}_6 - \mathcal{N}_{10}$ are related to machine assignment string. $\mathcal{N}_7, \mathcal{N}_9$ are the strategies for the bottleneck stage. $\mathcal{N}_1$ is the swapping of two randomly chosen π_i and π_j. $\mathcal{N}_2$ is used to generate solutions by inserting π_i into the position of π_j. $\mathcal{N}_3$ is shown below. Stochastically choose J_i, J_j, $a, b \in [1, L]$, $c, d \in [1, H]$, determine O^a_{ic}, O^b_{id}, O^a_{jc}, O^b_{jd} and their corresponding genes π_e, π_f, π_g, π_h, respectively, then swap π_e, π_f and exchange π_g, π_h on scheduling string. Taking Fig. 2 as an example, randomly select J_1, J_5, $a = 1$, $b = 2$, $c = 2$, $d = 3$, determine O^1_{12}, O^2_{13}, O^1_{52}, O^2_{53} and their corresponding genes $\pi_2 = 1$, $\pi_{27} = 1$, $\pi_7 = 5$, $\pi_{30} = 5$, then swap $\pi_2 = 1$ and $\pi_7 = 5$, and exchange $\pi_{27} = 1$ and $\pi_{30} = 5$.

$\mathcal{N}_4$ is show below. Stochastically select two genes π_j and π_k of J_i, and invert genes between them. $\mathcal{N}_5$ is described below. Randomly choose a job J_i, determine its corresponding $H \times L$ genes and delete them from scheduling string, then for each gene of J_i, insert the gene into a new randomly decided position k in scheduling string. For the example in Fig. 2, randomly select job J_3 and delete its all genes π_8, π_{11}, π_{14}, π_{20}, π_{23}, π_{26} from scheduling string, which becomes [1, 1, 1, 5, 2, 2, 5, 5, 4, 2, 2, 4, 2, 4, 1, 1, 2, 4, 4, 4, 1, 5, 5, 5], start with π_8, for each gene, insert it into a randomly chosen position on scheduling string, scheduling string finally becomes [3, 1, 3, 1, 1, 5, 3, 2, 2, 5, 5, 4, 2, 2, 3, 4, 3, 2, 4, 1, 1, 2, 3, 4, 4, 4, 1, 5, 5, 5].

$\mathcal{N}_6$ is shown as follows. Randomly select a machine q_{ig}, determine the processing stage k for this machine, $q_{ig} = h$, where h is stochastically chosen from $\{1, 2, \cdots, S_k\} \setminus \{q_{ig}\}$. $\mathcal{N}_7$ is similar to $\mathcal{N}_6$ expect that q_{ig} is the machine at bottleneck stage b. When $\mathcal{N}_8$ is executed, J_i and J_j are randomly selected, then $q_{i1}, q_{i2}, \cdots, q_{iH\times L}$ of J_i and $q_{j1}, q_{j2}, \cdots, q_{jH\times L}$ of J_j are swapped, respectively. $\mathcal{N}_9$ has the same steps as $\mathcal{N}_8$ expect that only swap machines at bottleneck stage b of J_i and J_j. $\mathcal{N}_{10}$ is shown as follows. Stochastically decided a job J_i, $w = 1$, repeat the following steps until $w > H \times L$: Perform $\mathcal{N}_6$ for q_{iw}, $w = w + 1$.

$\mathcal{N}_7, \mathcal{N}_9$ are proposed for the bottleneck stage due to the following feature of the problem: The new machine of a job J_i at the bottleneck stage b or the swap between machines at bottleneck stage b of J_i and J_j can significantly optimize the corresponding objective values with a high probability.

Multiple neighborhood search is executed in the following way. Let $t = 1$, repeat the following steps until $t > 10$: For solution x, produce a new solution $z \in \mathcal{N}_t(x)$, if $C_{\max}(z) < C_{\max}(x)$, replace x with z, $t = 11$; otherwise $t = t + 1$, where $\mathcal{N}_t(x)$ denotes the set of neighborhood solutions generated by $\mathcal{N}_t$ on x.

3.3 Class Evolution

Class evolution is composed of teacher competition, teacher's teaching and reinforcement search of elite class. Let $\Lambda = \{x^1_{teacher}, \cdots, x^{\alpha+\beta}_{teacher}\}$.

Teacher competition is described as follows:

1. For each teacher $x^i_{teacher} \in \Lambda$, stochastically select teacher $x^j_{teacher} \in \Lambda$, $i \neq j$, perform $GS(x^i_{teacher}, x^j_{teacher})$, and execute multiple neighborhood search on $x^i_{teacher} w$ times.
2. For each formal teacher $x^r_{teacher}$, $r = 1, 2, \cdots, \alpha$, let $t = \alpha + 1$, repeat the following steps until $t > \alpha + \beta$: If$C_{\max}\left(x^t_{teacher}\right) < C_{\max}\left(x^r_{teacher}\right)$, then swap $x^r_{teacher}$ and $x^t_{teacher} \in \Omega$, $t = t + 1$.

When $x^r_{teacher}$ and $x^t_{teacher} \in \Omega$ are swapped, let $x_{tmp} = x^r_{teacher}$, $\Omega = \Omega \setminus \{x^t_{teacher}\}$, $x^r_{teacher}$ is replaced with $x^t_{teacher}$, then x_{tmp} is added into $\Omega$and x_{tmp} becomes new $x^t_{teacher}$.

Teacher teaching is shown below. For each learner $x_i \in Cls_r$, perform $GS\left(x_i, x^r_{teacher}\right)$ and execute multiple neighborhood search on x_i, determine a learner $x_{worst} \in Cls_r$ with the biggest maximum completion time, randomly choose a substitute teacher $x^t_{teacher} \in \Omega$, and perform $GS(x_{worst}, x^t_{teacher})$.

Reinforcement search of elite class is performed in the following way. Sort all solutions in population P in ascending order of $C_{\max}^{x}$, and construct an elite class Cls^* with the best $\gamma \times N$ solutions; for each elite solution $x_i^* \in Cls^*$, randomly select another elite solution $x_j^* \in Cls^*$, perform $GS\left(x_i^*, x_j^*\right)$ and execute multiple neighborhood search w times on x_i^*, where $\gamma \times N > (\alpha + \beta)$.

Unlike the previous TLBO [40–43], ETLBO has reinforcement search of elite class used to substitute for learner phase. Since elite solutions are mostly composed of teachers and good learners, better solutions are more likely generated by global search and multiple neighborhood search on these elite solutions, and the waste of computational resources can be avoided on interactive learning between those worse learners with bigger $C_{\max}^{x}$.

3.4 Adaptive Adjustment on Teachers and Classes

Class quality is determined by the number of elite solutions in class. The quality Cqu_r of class Cls_r is defined as follows:

$$Cqu_r = |\{x_i \in Cls_r | x_i \in Cls^*\}| \tag{2}$$

Adaptive adjustment on teachers and classes is shown below:

(1) Sort all classes in descending order of Cqu_r, suppose that $Cqu_1 \geq Cqu_2 \geq \cdots \geq Cqu_\alpha$, $r = 1$, repeat the following steps until $r > (\alpha - 1)$, swap the best learner in Cls_r and the worst learner in Cls_{r+1}.

(2) For each solution $x_i \in P$, let $j = 1$, repeating the following steps until $j > (\alpha + \beta)$: If $C_{\max}(x_i) < C_{\max}\left(x_{teacher}^j\right)$ and $x_i \in Cls_r$, then swap $x_{teacher}^j$ and $x_i \in Cls_r$; if $C_{\max}(x_i) < C_{\max}\left(x_{teacher}^j\right)$ and $x_i \in \Lambda$, then swap $x_{teacher}^j$ and $x_i \in \Lambda$.

(3) Let $r = 1$ and Θ be empty, repeat the following steps until $r > \alpha$: for class Cls_r, select a teacher $x_{teacher}^i \in \Lambda$ by roulette selection [13], and swap $x_{teacher}^r$ and $x_{teacher}^i \in \Lambda$, then $\Lambda = \Lambda \backslash \left\{x_{teacher}^r\right\}$, $\Theta = \Theta \cup \left\{x_{teacher}^r\right\}$, $r = r + 1$.

(4) $\Omega = \Lambda$, $\Lambda = \Lambda \cup \Theta$.

When roulette selection is done, selection probability $prob_i = 1/C_{\max}^{x_{teacher}^i} / \sum_{x_{teacher}^j \in \Lambda} 1/C_{\max}^{x_{teacher}^j}$ is used.

In step (1), communication between classes Cls_r and Cls_{r+1} is done to avoid excessive differences among classes in solution quality. In step (2), the best learner can become teacher. In step (3), the formal teacher of each class is adjusted adaptively. Substitute teachers are updated in step (4). The above adaptive adjustment on learners and teachers can maintain high population diversity and make global search ability be effectively enhanced.

3.5 Algorithm Description

The search procedure of ETLBO is shown below:

1. Randomly produce an initial population P with N solutions and divide population into α classes.
2. Execute teacher competition.
3. Perform teacher's teaching.
4. Implement reinforcement search of elite class.
5. Apply adaptive adjustment on teachers and classes.
6. If the termination condition is not met, go to step (2); otherwise, stop search and output the optimum solution.

Fig. 3 describes flow chart of ETLBO.

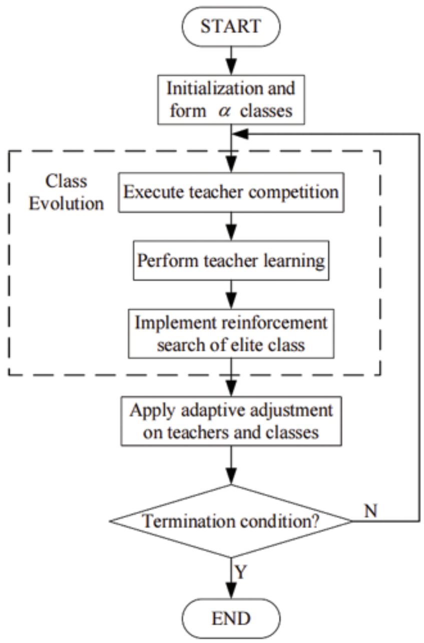

Figure 3: Flow chart of ETLBO

ETLBO has the following new features. Teachers are divided into formal ones and substitute ones. Teacher competition is applied between formal and substitute teachers. Teacher's teaching is performed and reinforcement search of elite class is used to replace learner phase. Adaptive adjustment on teachers and classes is conducted based on class quality assessment. These features promote a balance between exploration and exploitation, then good results can finally be obtained.

4 Computational Experiments

Extensive experiments are conducted to test the performance of ETLBO for RHFSP with bottleneck stage. All experiments are implemented by using Microsoft Visual C++ 2022 and run on i7-8750H CPU (2.20 GHz) and 24 GB RAM.

4.1 Test Instance and Comparative Algorithms

60 instances are randomly produced. For each instance depicted by $n \times H \times L$, where $L \in \{2, 3\}$, $n \in [10,100]$, $H \in \{3, 4, 5\}$. If $H = 3$, then $b = 2$, $S_b = 4$; if $H = 4$, then $b = 3$, $S_b = 5$; if $H = 5$, then $b = 4$, $S_b = 6$; if $k \neq b$, $S_k \in [2, 4]$, $p_{ik} \in [10, 20]$; otherwise, $p_{ik} \in [200, 300]$. S_k and p_{ik} are integer and follow uniform distribution within their intervals.

For the considered RHFSP with maximum completion time minimization, there are no existing methods. To demonstrate the advantages of ETLBO for the RHFSP with bottleneck stage, hybrid harmony search and genetic algorithm (HHSGA, [12]), improved dual-population genetic algorithm (IDPGA, [13]) and discrete differential evolution algorithm (DDE, [14]) are selected as comparative algorithms.

Lin et al. [12] proposed an algorithm named HHSGA for RHFSP with limited buffer to minimize weighted values of maximum completion time and mean flowtime. Tang et al. [13] designed IDPGA to solve RHFSP with missing operations to minimize maximum completion time and energy consumption. Zhang et al. [14] applied DDE to address RHFSP with machine eligibility to minimize total tardiness. These algorithms have been successfully applied to deal with RHFSP, so they can be directly used to handle the considered RHFSP by incorporating bottleneck formation into the decoding process; moreover, missing judgment vector and related operators of IDPGA are removed.

A TLBO is constructed, it consists of a class in which the best solution be seen as a teacher and remaining solutions are students, and it includes a teacher phase and a learner phase. Teacher phase is implemented by each learner learning from the teacher and learner phase is done by interactive learning between a learner and another randomly selected learner. These activities are the same as global search in ETLBO. The comparisons between ETLBO and TLBO are applied to show the effect of new strategies of ETLBO.

4.2 Parameter Settings

It can be found that ETLBO can converge well when $0.5 \times n \times H \times L$ seconds CPU time reaches; moreover, when $0.5 \times n \times H \times L$ seconds CPU time is applied, HHSGA, IDPGA, DDE, and TLBO also converge fully within this CPU time, so this time is chosen as stopping condition.

Other parameters of ETLBO, namely N, α, β, γ, and w, are tested by using Taguchi method [44] on instance $50 \times 4 \times 2$. Table 2 shows the levels of each parameter. ETLBO with each combination runs 10 times independently for the chosen instance.

Table 2: Level of the parameters

Parameter	Factor level			
	1	2	3	4
N	30	40	50	60
α	2	3	4	5
β	1	2	3	4
γ	0.1	0.2	0.3	0.4
w	1	2	3	4

Fig. 4 shows the results of MIN and S/N ratio, which is defined as $-10 \times \log_{10}\left(\mathrm{MIN}^2\right)$ and MIN is the best solution in 10 runs. It can be found in Fig. 4 that ETLBO with following combination $N = 50$, $\alpha = 3$, $\beta = 2$, $\gamma = 0.2$, $w = 2$ can obtain better results than ETLBO with other combinations, so above combination is adopted.

TLBO have N and stopping condition are given with the same settings as ETLBO. All parameters of HHSGA, IDPGA and DDE except stopping condition are obtained directly from [12–14]. The experimental results show that those settings of each comparative algorithm are still effective and comparative algorithms with those settings can produce better results than HHSGA, IDPGA and DDE with other settings.

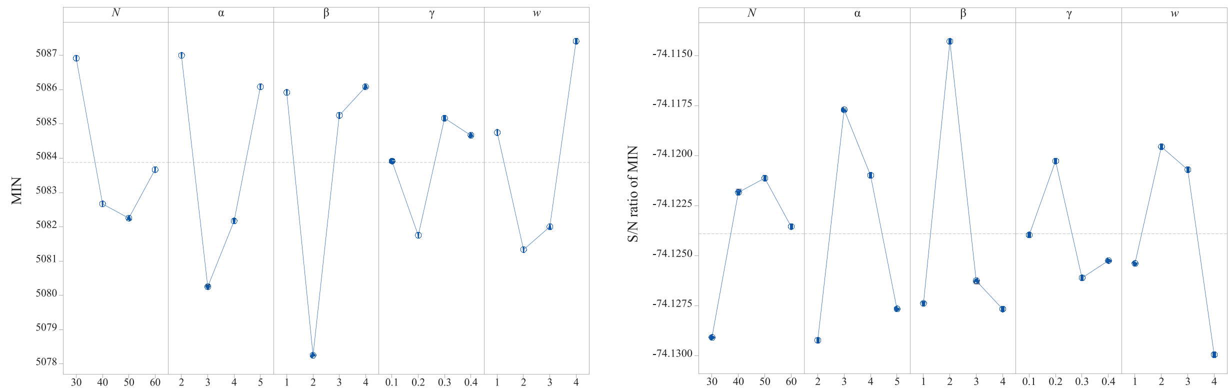

Figure 4: Main effect plot for mean MIN and S/N ratio

4.3 Result and Discussions

ETLBO is compared with HHSGA, IDPGA, DDE and TLBO. Each algorithm randomly runs 10 times for each instance. AVG, STD denotes the average and standard deviation of solutions in 10 run times. Tables 3–5 describe the corresponding results of five algorithms. Figs. 5 and 6 show box plots of all algorithms and convergence curves of instance $50 \times 3 \times 3$ and $70 \times 5 \times 2$. The relative percentage deviation (RPD) between the best performs algorithm and other four algorithms is used in Fig. 5. RPD_{MIN}, RPD_{AVG}, RPD_{STD} are defined:

$$RPD_{MIN} = \frac{MIN - MIN^*}{MIN^*} \times 100\% \tag{3}$$

where MIN^* (MAX^*, STD^*) is the smallest MIN (MAX, STD) obtained by all algorithms, when MIN and MIN^* are replaced with STD(AVG) and STD^*(AVG^*), respectively, RPD_{STD} (RPD_{AVG}) is obtained in the same way.

Table 3: Computational results of five algorithms on MIN

$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO	$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO
$10 \times 3 \times 2$	**1221**	1221	1221	1221	1221	$60 \times 3 \times 2$	**7426**	7444	7453	7563	7498
$10 \times 3 \times 3$	**1903**	1903	1905	1909	1905	$60 \times 3 \times 3$	**11346**	11404	11413	11472	11435
$10 \times 4 \times 2$	**1069**	1069	1069	1069	1069	$60 \times 4 \times 2$	**6019**	6047	6055	6101	6088
$10 \times 4 \times 3$	**1549**	1558	1556	1558	1553	$60 \times 4 \times 3$	**9207**	9251	9267	9298	9301
$10 \times 5 \times 2$	**955**	955	955	955	955	$60 \times 5 \times 2$	**5141**	5172	5168	5212	5171
$10 \times 5 \times 3$	**1497**	1499	1499	1511	1499	$60 \times 5 \times 3$	**7824**	7866	7854	7998	7933
$20 \times 3 \times 2$	**2571**	2577	2583	2597	2587	$70 \times 3 \times 2$	**8694**	8733	8744	8816	8787
$20 \times 3 \times 3$	**3993**	4001	4010	4022	4015	$70 \times 3 \times 3$	**13386**	13469	13483	13559	13505
$20 \times 4 \times 2$	**2053**	2062	2068	2079	2073	$70 \times 4 \times 2$	**7079**	7099	7122	7175	7144
$20 \times 4 \times 3$	**3169**	3188	3176	3202	3182	$70 \times 4 \times 3$	**10306**	10349	10362	10444	10395
$20 \times 5 \times 2$	**1737**	1743	1752	1767	1762	$70 \times 5 \times 2$	**6054**	6073	6077	6118	6083
$20 \times 5 \times 3$	**2648**	2671	2669	2685	2677	$70 \times 5 \times 3$	**9388**	9441	9464	9598	9500
$30 \times 3 \times 2$	**3963**	3999	4003	4054	4049	$80 \times 3 \times 2$	**9961**	9994	10009	10132	10052
$30 \times 3 \times 3$	**5642**	5690	5713	5812	5764	$80 \times 3 \times 3$	**15393**	15434	15489	15673	15583
$30 \times 4 \times 2$	**3057**	3088	3073	3123	3094	$80 \times 4 \times 2$	**7912**	7961	7998	8127	8024
$30 \times 4 \times 3$	**4771**	4798	4801	4879	4833	$80 \times 4 \times 3$	**12097**	12134	12186	12363	12240
$30 \times 5 \times 2$	**2678**	2689	2701	2746	2735	$80 \times 5 \times 2$	**7088**	7097	7122	7287	7207

(Continued)

Table 3 (continued)

$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO	$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO
30 × 5 × 3	**4037**	4058	4069	4132	4101	80 × 5 × 3	**10114**	10137	10166	10345	10231
40 × 3 × 2	**5152**	5198	5213	5348	5259	90 × 3 × 2	**11659**	11693	11719	11926	11832
40 × 3 × 3	**7493**	7533	7523	7623	7577	90 × 3 × 3	**16951**	16992	17041	17221	17148
40 × 4 × 2	**4127**	4136	4147	4200	4188	90 × 4 × 2	**9083**	9120	9138	9292	9237
40 × 4 × 3	**6125**	6140	6153	6278	6196	90 × 4 × 3	**13714**	13772	13789	13990	13881
40 × 5 × 2	**3453**	3477	3482	3591	3514	90 × 5 × 2	**7774**	7801	7811	7976	7893
40 × 5 × 3	**5235**	5246	5258	5345	5306	90 × 5 × 3	**11900**	11947	11998	12203	12121
50 × 3 × 2	**6244**	6277	6268	6351	6293	100 × 3 × 2	**12373**	12408	12443	12765	12555
50 × 3 × 3	**9237**	9277	9255	9303	9279	100 × 3 × 3	**18570**	18611	18687	18934	18776
50 × 4 × 2	**5054**	5099	5086	5127	5111	100 × 4 × 2	**10162**	10207	10242	10439	10365
50 × 4 × 3	**7581**	7601	7613	7662	7654	100 × 4 × 3	**14992**	15038	15080	15298	15142
50 × 5 × 2	**4331**	4370	4384	4438	4424	100 × 5 × 2	**8698**	8742	8763	8878	8844
50 × 5 × 3	**6773**	6788	6815	6907	6872	100 × 5 × 3	**13066**	13133	13176	13364	13223

Table 4: Computational results of five algorithms on AVG

$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO	$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO
10 × 3 × 2	**1225**	1229	1227	1231	1231	60 × 3 × 2	**7430**	7471	7488	7627	7553
10 × 3 × 3	**1913**	1922	1922	1926	1916	60 × 3 × 3	**11355**	11483	11446	11574	11523
10 × 4 × 2	**1073**	1078	1085	1089	1076	60 × 4 × 2	**6034**	6098	6101	6153	6129
10 × 4 × 3	**1576**	1578	1585	1590	1580	60 × 4 × 3	**9235**	9303	9326	9389	9374
10 × 5 × 2	966	968	**964**	977	968	60 × 5 × 2	**5200**	5225	5219	5292	5214
10 × 5 × 3	1543	1545	**1535**	1554	1537	60 × 5 × 3	**7919**	7940	7958	8098	8009
20 × 3 × 2	**2573**	2598	2610	2622	2612	70 × 3 × 2	**8698**	8757	8759	8902	8846
20 × 3 × 3	**3999**	4040	4059	4048	4054	70 × 3 × 3	**13395**	13505	13511	13668	13589
20 × 4 × 2	**2064**	2088	2091	2101	2096	70 × 4 × 2	**7090**	7147	7160	7305	7222
20 × 4 × 3	**3180**	3215	3211	3291	3216	70 × 4 × 3	**10318**	10388	10407	10562	10496
20 × 5 × 2	**1758**	1763	1779	1794	1786	70 × 5 × 2	**6083**	6104	6111	6198	6116
20 × 5 × 3	**2715**	2728	2716	2732	2735	70 × 5 × 3	**9448**	9518	9545	9704	9613
30 × 3 × 2	**3966**	4053	4032	4085	4095	80 × 3 × 2	**9964**	10039	10074	10241	10128
30 × 3 × 3	**5647**	5729	5754	5886	5813	80 × 3 × 3	**15400**	15523	15572	15803	15680
30 × 4 × 2	**3062**	3123	3112	3191	3136	80 × 4 × 2	**7926**	7998	8045	8219	8078
30 × 4 × 3	**4779**	4850	4838	4951	4871	80 × 4 × 3	**12119**	12220	12258	12555	12366
30 × 5 × 2	**2717**	2729	2755	2797	2787	80 × 5 × 2	**7119**	7148	7160	7376	7306
30 × 5 × 3	**4096**	4133	4117	4200	4179	80 × 5 × 3	**10179**	10206	10231	10439	10316
40 × 3 × 2	**5155**	5234	5275	5424	5305	90 × 3 × 2	**11667**	11744	11779	12099	12013
40 × 3 × 3	**7499**	7577	7557	7664	7640	90 × 3 × 3	**16963**	17088	17129	17400	17246
40 × 4 × 2	**4135**	4174	4187	4271	4218	90 × 4 × 2	**9101**	9172	9183	9385	9332
40 × 4 × 3	**6132**	6186	6184	6354	6284	90 × 4 × 3	**13738**	13852	13927	14100	13983
40 × 5 × 2	**3475**	3530	3507	3653	3565	90 × 5 × 2	**7838**	7886	7891	8074	7933
40 × 5 × 3	**5272**	5314	5301	5407	5378	90 × 5 × 3	**11973**	12069	12116	12375	12286
50 × 3 × 2	**6249**	6310	6329	6430	6340	100 × 3 × 2	**12382**	12448	12504	12864	12720
50 × 3 × 3	**9242**	9316	9319	9413	9351	100 × 3 × 3	**18586**	18705	18797	19194	18939
50 × 4 × 2	**5067**	5139	5136	5209	5214	100 × 4 × 2	**10193**	10275	10357	10546	10481
50 × 4 × 3	**7588**	7651	7641	7727	7740	100 × 4 × 3	**15016**	15106	15186	15387	15333
50 × 5 × 2	**4350**	4404	4410	4501	4469	100 × 5 × 2	**8791**	8818	8859	8974	8892
50 × 5 × 3	6839	**6832**	6868	6976	6949	100 × 5 × 3	**13130**	13220	13239	13492	13308

Table 5: Computational results of five algorithms on STD

$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO	$n \times H \times L$	ETLBO	HHSGA	IDPGA	DDE	TLBO
10 × 3 × 2	**2.00**	6.39	4.40	5.85	6.68	60 × 3 × 2	**4.50**	19.43	30.06	47.26	37.49
10 × 3 × 3	**7.99**	10.56	12.13	15.23	9.62	60 × 3 × 3	**5.96**	54.03	43.35	79.62	48.45
10 × 4 × 2	**4.86**	7.03	9.79	15.28	8.45	60 × 4 × 2	**8.48**	33.53	33.84	47.58	36.98
10 × 4 × 3	17.47	**12.07**	20.54	23.61	19.15	60 × 4 × 3	**18.41**	29.21	33.54	50.54	55.34
10 × 5 × 2	**8.30**	9.66	9.58	17.09	10.73	60 × 5 × 2	31.62	**28.35**	42.83	71.69	38.58
10 × 5 × 3	27.80	**27.78**	34.06	28.74	32.75	60 × 5 × 3	47.58	**44.49**	62.16	85.27	58.07
20 × 3 × 2	**2.47**	17.84	18.23	15.97	18.71	70 × 3 × 2	**3.80**	16.77	17.24	48.61	43.59
20 × 3 × 3	**4.11**	35.32	28.20	25.52	27.38	70 × 3 × 3	**5.71**	15.06	21.37	69.44	57.63
20 × 4 × 2	**7.29**	18.86	17.58	19.87	22.64	70 × 4 × 2	**9.20**	24.70	23.28	88.01	48.35
20 × 4 × 3	**12.78**	17.87	20.20	52.55	24.76	70 × 4 × 3	**11.50**	25.14	32.57	72.62	65.53
20 × 5 × 2	13.88	**12.52**	16.43	19.27	22.03	70 × 5 × 2	**17.10**	24.16	27.27	73.68	36.15
20 × 5 × 3	**27.57**	43.10	28.55	42.18	32.31	70 × 5 × 3	55.09	55.83	**44.08**	63.67	80.07
30 × 3 × 2	**1.89**	33.05	27.35	22.55	31.42	80 × 3 × 2	**3.11**	30.69	43.18	80.53	44.60
30 × 3 × 3	**4.83**	32.58	22.73	60.86	45.99	80 × 3 × 3	**5.03**	51.79	55.66	95.22	65.15
30 × 4 × 2	**3.28**	16.93	20.95	40.70	26.15	80 × 4 × 2	**9.90**	32.95	28.90	68.05	46.91
30 × 4 × 3	**4.06**	37.14	28.17	38.60	36.37	80 × 4 × 3	**19.80**	61.11	45.45	111.46	85.82
30 × 5 × 2	**21.23**	30.92	34.57	54.50	36.31	80 × 5 × 2	**24.81**	35.46	27.12	65.59	64.91
30 × 5 × 3	**29.62**	41.00	32.08	57.23	47.63	80 × 5 × 3	44.73	54.56	**40.46**	102.65	52.79
40 × 3 × 2	**2.87**	26.71	30.16	51.61	29.57	90 × 3 × 2	**7.67**	37.74	27.16	114.09	138.62
40 × 3 × 3	**5.11**	32.05	22.63	35.03	36.49	90 × 3 × 3	**8.06**	73.98	56.87	107.96	55.88
40 × 4 × 2	**4.91**	31.69	27.66	45.43	29.75	90 × 4 × 2	**10.02**	32.49	30.61	74.26	68.74
40 × 4 × 3	**4.79**	32.60	24.42	56.59	41.25	90 × 4 × 3	**17.80**	58.56	56.90	94.36	89.35
40 × 5 × 2	**11.60**	38.87	16.66	42.31	36.29	90 × 5 × 2	35.49	58.64	51.41	72.34	**31.37**
40 × 5 × 3	27.22	28.18	**23.62**	60.80	62.14	90 × 5 × 3	**56.95**	85.74	94.86	123.54	88.36
50 × 3 × 2	**4.03**	28.33	34.38	45.28	37.28	100 × 3 × 2	**4.63**	31.14	62.92	83.09	110.50
50 × 3 × 3	**4.22**	17.21	44.58	63.72	48.15	100 × 3 × 3	**5.17**	53.91	79.85	133.47	100.55
50 × 4 × 2	**8.37**	26.89	34.94	48.41	70.07	100 × 4 × 2	**16.87**	58.85	45.21	85.70	74.25
50 × 4 × 3	**7.19**	28.21	23.89	40.85	71.48	100 × 4 × 3	**14.83**	58.77	77.58	65.65	126.99
50 × 5 × 2	**13.41**	26.61	30.73	50.56	36.55	100 × 5 × 2	54.28	53.29	71.18	71.06	**47.01**
50 × 5 × 3	83.29	**35.29**	37.97	56.06	42.95	100 × 5 × 3	**34.88**	59.36	58.12	69.55	89.60

Table 6 describes the results of a pair-sample t-test, in which t-test (A1, A2) means that a paired t-test is performed to judge whether algorithm A1 gives a better sample mean than A2. If the significance level is set at 0.05, a statistically significant difference between A1 and A2 is indicated by a p-value less than 0.05.

As shown in Tables 3–5, ETLBO obtains smaller MIN than TLBO on all instances and MIN of ETLBO is lower than that of TLBO by at least 50 on 46 instances. It can be found from Table 4 that AVG of ETLBO is better than that of TLBO on 59 of 60 instances and SFLA is worse AVG than ETLBO by at least 50 on 45 instances. Table 5 shows that ETLBO obtains smaller STD than TLBO on 58 instances. Table 6 shows that there are notable performance differences between ETLBO and TLBO in a statistical sense. Fig. 5 depicts the notable differences between STD of the two algorithms, and Fig. 6 reveals that ETLBO significantly converges better than TLBO. It can be concluded that teacher competition, reinforcement search of elite class and adaptive adjustment on teachers and classes have a positive impact on the performance of ETLBO.

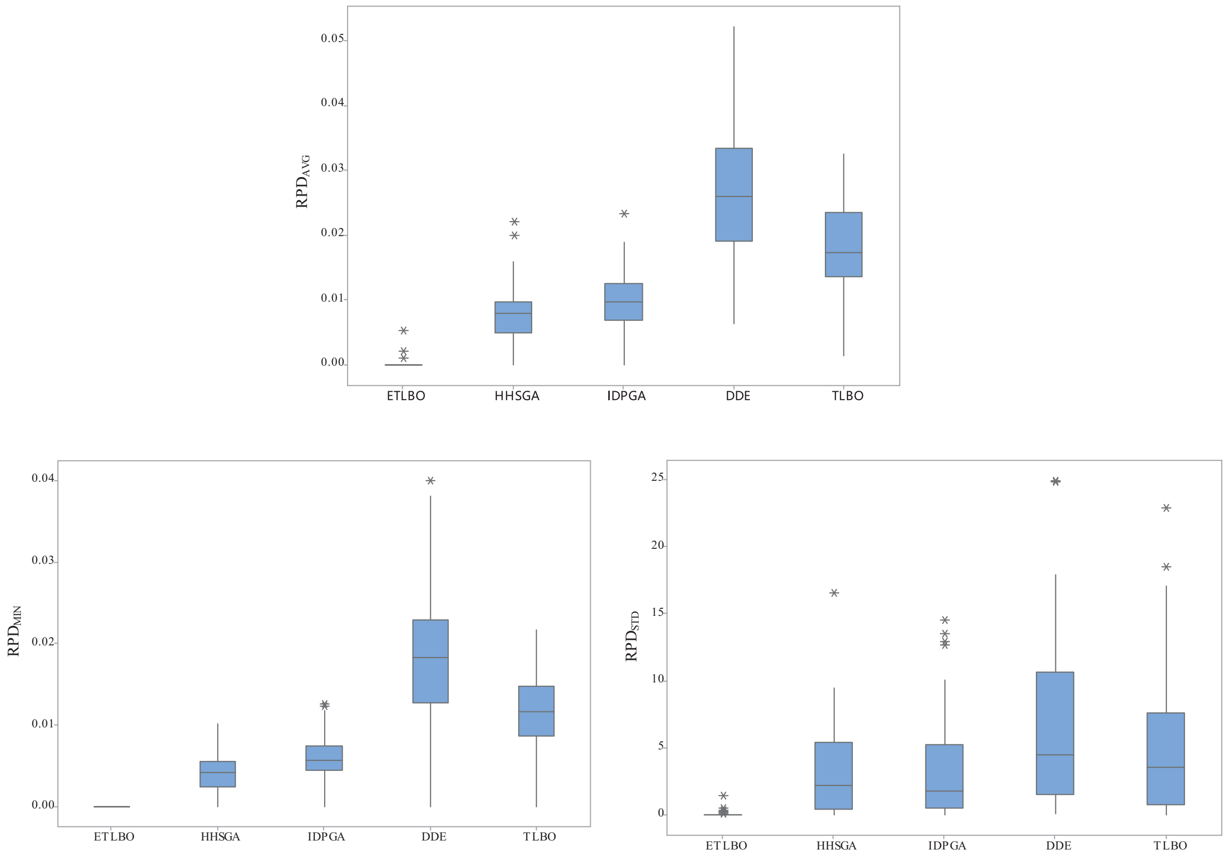

Figure 5: Box plots for all algorithms

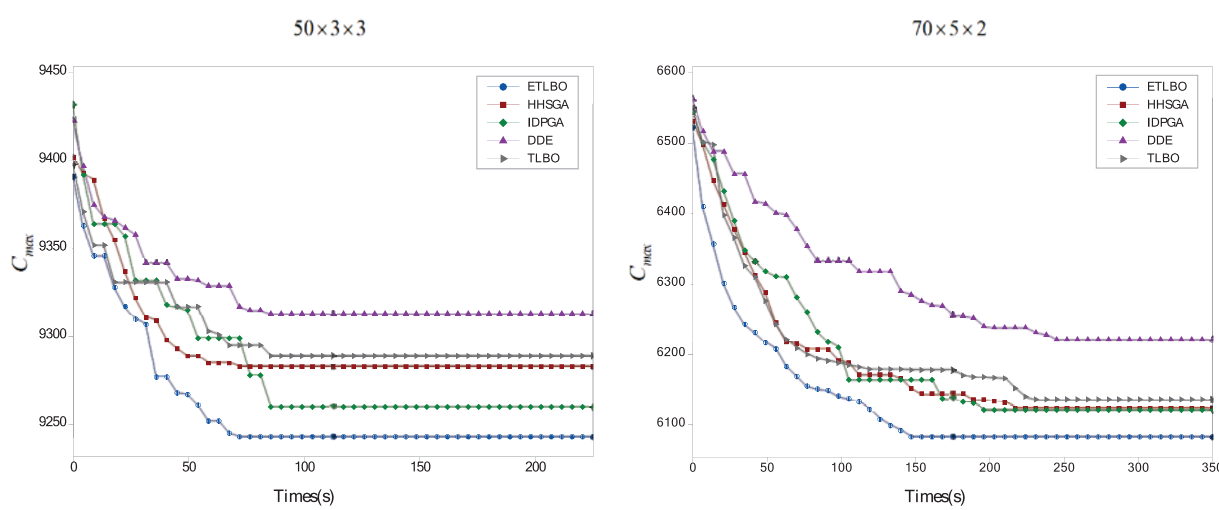

Figure 6: Convergence curves of instance $50 \times 3 \times 3$ and $70 \times 5 \times 2$

Table 6 : t-test result of the algorithm

t-text	p-MIN	p-AVG	p-STD
t-text (ETLBO, HHSGA)	0.000	0.000	0.000
t-text (ETLBO, IDPGA)	0.000	0.000	0.000
t-text (ETLBO, DDE)	0.000	0.000	0.000
t-text (ETLBO, TLBO)	0.000	0.000	0.000

Table 3 describes that ETLBO performs better than HHSGA and IDPGA on MIN for all instances. As can be seen from Table 4, ETLBO produces smaller AVG than with the two comparative algorithms on 57 of 60 instances; moreover, AVG of ETLBO is less than that of HHSGA by at least 50 on 26 instances and IDPGA by at least 50 on 48 instances. Table 5 also shows that ETLBO obtains smaller STD than the two comparative algorithms on 49 instances. ETLBO converges better than HHSGA and IDPGA. The results in Table 6, Figs. 5 and 6 also demonstrate the convergence advantage of ETLBO.

It can be concluded from Tables 3–5 that ETLBO performs significantly better than DDE. ETLBO produces smaller MIN than DDE in all instances, also generates better AVG than DDE by at least 50 on 45 instances and obtains better STD than or the same STD as DDE on nearly all instances. ETLBO performs notably better than DDE, and the same conclusion can be found in Table 6. Fig. 5 illustrates the significant difference in STD, and Fig. 6 demonstrates the notable convergence advantage of ETLBO.

As analyzed above, ETLBO outperforms its comparative algorithms. The good performance of ETLBO mainly results from its teacher competition, reinforcement search of elite class and adaptive adjustment on teachers and classes. Teacher competition is proposed to make full use of teacher solutions, reinforcement search of elite class performs more searches for better solutions to avoid wasting computational resources, adaptive adjustment on teachers and classes dynamically adjusts class composition according to class quality, as a result, which can effectively prevent the algorithm from falling into local optima. Thus, it can be concluded that ETLBO is a promising method for RHFSP with bottleneck stage.

5 Conclusion and Future Work

This study considers RHFSP with bottleneck stage, and a new algorithm named ETLBO is presented to minimize maximum completion time. In ETLBO, teachers are divided into formal teachers and substitute teachers. A new teacher phase is implemented, which includes two types of teachers' competition and teaching phases. Reinforcement search of the elite class is used to replace the learner phase. Based on class quality, adaptive adjustment is made for classes and teachers to change the composition of them. The experimental results show that ETLBO is a very competitive algorithm for the considered RHFSP.

In the near future, we will continue to focus on RHFSP and use other meta-heuristics such as artificial bee colony algorithm and imperialist competitive algorithm to solve it. Some new optimization mechanisms, such as cooperation and reinforcement learning, are added into meta-heuristics are our future research topics. Fuzzy RHFSP and distributed RHFSP are another of our directions. Furthermore, the application of ETLBO to deal with other scheduling problems is also worthy of further investigation.

Acknowledgement: The author would like to thank the editors and reviewers for their valuable work, as well as the supervisor and family for their valuable support during the research process.

Funding Statement: This research was funded by the National Natural Science Foundation of China (Grant Number 61573264).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Deming Lei, Surui Duan; data collection: Surui Duan; analysis and interpretation of results: Deming Lei, Surui Duan, Mingbo Li, Jing Wang; draft manuscript preparation: Deming Lei, Surui Duan, Mingbo Li, Jing Wang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data supporting this study are described in the first paragraph of Section 4.1.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Low, C. J. Hsu and C. T. Su, "A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines," *Comput. Oper. Res.*, vol. 35, no. 3, pp. 845–853, Mar. 2008. doi: 10.1016/j.cor.2006.04.004.

[2] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, Aug. 2010. doi: 10.1016/j.ejor.2009.09.024.

[3] J. Dong, and C. Ye, "Green scheduling of distributed two-stage reentrant hybrid flow shop considering distributed energy resources and energy storage system," *Comput. Ind. Eng.*, vol. 169, pp. 108146, Mar. 2022. doi: 10.1016/j.cie.2022.108146.

[4] J. S. Chen, "A branch and bound procedure for the reentrant permutation flow-shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 29, pp. 1186–1193, Aug. 2006. doi: 10.1007/s00170-005-0017-x.

[5] J. C. H. Pan and J. S. Chen, "Minimizing makespan in re-entrant permutation flow-shops," *J. Oper. Res. Soc.*, vol. 54, pp. 642–653, Jun. 2003. doi: 10.1057/palgrave.jors.2601556.

[6] S. Kumar and M. K. Omar, "Stochastic re-entrant line modeling for an environment stress testing in a semiconductor assembly industry," *Appl. Math. Comput.*, vol. 173, no. 1, pp. 603–615, Feb. 2006. doi: 10.1016/j.amc.2005.04.050.

[7] Y. H. Lee and B. Lee, "Push-pull production planning of the re-entrant process," *Int. J. Adv. Manuf. Technol.*, vol. 22, pp. 922–931, Aug. 2003. doi: 10.1007/s00170-003-1653-7.

[8] F. Xu *et al.*, "Research on green reentrant hybrid flow shop scheduling problem based on improved moth-flame optimization algorithm," *Process.*, vol. 10, no. 12, pp. 2475, Nov. 2022. doi: 10.3390/pr10122475.

[9] B. H. Zhou, L. M. Hu and Z. Y. Zhong, "A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem," *Neural. Comput. Appl.*, vol. 30, no. 1, pp. 193–209, Jul. 2018. doi: 10.1007/s00521-016-2692-y.

[10] H. M. Cho *et al.*, "Bi-objective scheduling for reentrant hybrid flow shop using Pareto genetic algorithm," *Comput. Ind. Eng.*, vol. 61, no. 3, pp. 529–541, Oct. 2011. doi: 10.1016/j.cie.2011.04.008.

[11] J. N. Shen, L. Wang and H. Y. Zheng, "A modified teaching-learning-based optimisation algorithm for bi-objective re-entrant hybrid flowshop scheduling," *Int. J. Prod. Res.*, vol. 54, no. 12, pp. 3622–3639, Jun. 2016. doi: 10.1080/00207543.2015.1120900.

[12] C. C. Lin, W. Y. Liu, and Y. H. Chen, "Considering stockers in reentrant hybrid flow shop scheduling with limited buffer capacity," *Comput. Ind. Eng.*, vol. 139, pp. 106154, Jan. 2020. doi: 10.1016/j.cie.2019.106154.

[13] H. T. Tang *et al.*, "Hybrid flow-shop scheduling problems with missing and re-entrant operations considering process scheduling and production of energy consumption," *Sustainability*, vol. 15, no. 10, pp. 7982, May 2023. doi: 10.3390/su15107982.

[14] X. Y. Zhang and L. Chen, "A re-entrant hybrid flow shop scheduling problem with machine eligibility constraints," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5293–5305, Dec. 2017. doi: 10.1080/00207543.2017.1408971.

[15] C. Chamnanlor *et al.*, "Embedding ant system in genetic algorithm for re-entrant hybrid flow shop scheduling problems with time window constraints," *J. Intell. Manuf.*, vol. 28, pp. 1915–1931, Dec. 2017. doi: 10.1007/s10845-015-1078-9.

[16] X. L. Wu and Z. Cao, "An improved multi-objective evolutionary algorithm based on decomposition for solving re-entrant hybrid flow shop scheduling problem with batch processing machines," *Comput. Ind. Eng.*, vol. 169, pp. 108236, Jul. 2022. doi: 10.1016/j.cie.2022.108236.

[17] A. Costa, V. Fernandez-Viagas, and J. M. Framiñan, "Solving the hybrid flow shop scheduling problem with limited human resource constraint," *Comput. Ind. Eng.*, vol. 146, pp. 106545, Aug. 2020. doi: 10.1016/j.cie.2020.106545.

[18] W. Shao, Z. Shao, and D. Pi, "Modelling and optimization of distributed heterogeneous hybrid flow shop lot-streaming scheduling problem," *Expert. Syst. Appl.*, vol. 214, pp. 119151, Mar. 2023. doi: 10.1016/j.eswa.2022.119151.

[19] C. J. Liao, E. Tjandradjaja, and T. P. Chung, "An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem," *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1755–1764, Jun. 2012. doi: 10.1016/j.asoc.2012.01.011.

[20] C. Zhang *et al.*, "A discrete whale swarm algorithm for hybrid flow-shop scheduling problem with limited buffers," *Robot. CIM-Int. Manuf.*, vol. 68, pp. 102081, Oct. 2020. doi: 10.1016/j.rcim.2020.102081.

[21] J. Wang, D. Lei, and H. Tang, "An adaptive artificial bee colony for hybrid flow shop scheduling with batch processing machines in casting process," *Int. J. Prod. Res.*, pp. 1–16, Nov. 2023. doi: 10.1080/00207543.2023.2279145.

[22] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput. Aided. Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011. doi: 10.1016/j.cad.2010.12.015.

[23] M. K. Sun, Z. Y. Cai, and H. A. Zhang, "A teaching-learning-based optimization with feedback for L-R fuzzy flexible assembly job shop scheduling problem with batch splitting," *Expert. Syst. Appl.*, vol. 224, pp. 120043, Aug. 2023. doi: 10.1016/j.eswa.2023.120043.

[24] A. Baykasoğlu, A. Hamzadayi, and S. Y. Köse, "Testing the performance of teaching-learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases," *Inf. Sci.*, vol. 276, pp. 204–218, Aug. 2014. doi: 10.1016/j.ins.2014.02.056.

[25] Z. Xie *et al.*, "An effective hybrid teaching-learning-based optimization algorithm for permutation flow shop scheduling problem," *Adv. Eng. Softw.*, vol. 77, pp. 35–47, Nov. 2014. doi: 10.1016/j.advengsoft.2014.07.006.

[26] W. Shao, D. Pi, and Z. Shao, "An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem," *Appl. Soft Comput.*, vol. 61, pp. 193–210, Dec. 2017. doi: 10.1016/j.asoc.2017.08.020.

[27] C. Song, "A hybrid multi-objective teaching-learning based optimization for scheduling problem of hybrid flow shop with unrelated parallel machine," *IEEE Access*, vol. 9, pp. 56822–56835, Apr. 2021. doi: 10.1109/ACCESS.2021.3071729.

[28] R. Buddala and S. S. Mahapatra, "Improved teaching-learning-based and JAYA optimization algorithms for solving flexible flow shop scheduling problems," *J. Ind. Eng. Int.*, vol. 14, pp. 555–570, Sep. 2018. doi: 10.1007/s40092-017-0244-4.

[29] B. J. Xi and D. M. Lei, "Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time," *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 113–129, Jun. 2022. doi: 10.23919/CSMS.2022.0002.

[30] U. Balande and D. Shrimankar, "A modified teaching learning metaheuristic algorithm with opposite-based learning for permutation flow-shop scheduling problem," *Evol. Intell.*, vol. 15, no. 1, pp. 57–79, Mar. 2022. doi: 10.1007/s12065-020-00487-5.

[31] X. Ji *et al.*, "An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry," *Appl. Soft Comput.*, vol. 57, pp. 504–516, Aug. 2017. doi: 10.1016/j.asoc.2017.04.029.

[32] R. Buddala and S. S. Mahapatra, "An integrated approach for scheduling flexible job-shop using teaching-learning-based optimization method," *J. Ind. Eng. Int.*, vol. 15, no. 1, pp. 181–192, Mar. 2019. doi: 10.1007/s40092-018-0280-8.

[33] M. Rostami and A. Yousefzadeh, "A gamified teaching-learning based optimization algorithm for a three-echelon supply chain scheduling problem in a two-stage assembly flow shop environment," *Appl. Soft Comput.*, vol. 146, pp. 110598, Oct. 2023. doi: 10.1016/j.asoc.2023.110598.

[34] R. Buddala and S. S. Mahapatra, "Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown," *Int. J. Adv. Manuf. Technol.*, vol. 100, pp. 1419–1432, Feb. 2019. doi: 10.1007/s00170-018-2805-0.

[35] J. Jayanthi *et al.*, "Segmentation of brain tumor magnetic resonance images using a teaching-learning optimization algorithm," *Comput. Mater. Contin.*, vol. 68, no. 3, pp. 4191–4203, Mar. 2021. doi: 10.32604/cmc.2021.012252.

[36] D. M. Lei, B. Su, and M. Li, "Cooperated teaching-learning-based optimisation for distributed two-stage assembly flow shop scheduling," *Int. J. Prod. Res.*, vol. 59, no. 23, pp. 7232–7245, Nov. 2020. doi: 10.1080/00207543.2020.1836422.

[37] D. M. Lei and B. Su, "A multi-class teaching-learning-based optimization for multi-objective distributed hybrid flow shop scheduling," *Knowl. Based. Syst.*, vol. 263, pp. 110252, Jan. 2023. doi: 10.1016/j.knosys.2023.110252.

[38] A. Dubey, U. Gupta, and S. Jain, "Medical data clustering and classification using TLBO and machine learning algorithms," *Comput. Mater. Contin.*, vol. 70, no. 3, pp. 4523–4543, Oct. 2021. doi: 10.32604/cmc.2022.021148.

[39] D. M. Lei and B. J. Xi, "Diversified teaching-learning-based optimization for fuzzy two-stage hybrid flow shop scheduling with setup time," *J. Intell. Fuzzy Syst.*, vol. 41, no. 2, pp. 4159–4173, Sep. 2021. doi: 10.3233/JIFS-210764.

[40] D. M. Lei, L. Gao, and Y. L. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Trans. Eng. Manage.*, vol. 65, no. 2, pp. 330–340, May. 2018. doi: 10.1109/TEM.2017.2774281.

[41] A. Sharma *et al.*, "Identification of photovoltaic module parameters by implementing a novel teaching learning based optimization with unique exemplar generation scheme (TLBO-UEGS)," *Energy Rep.*, vol. 10, pp. 1485–1506, Nov. 2023. doi: 10.1016/j.egyr.2023.08.019.

[42] M. Arashpour *et al.*, "Predicting individual learning performance using machine-learning hybridized with the teaching-learning-based optimization," *Comput. Appl. Eng. Educ.*, vol. 31, no. 1, pp. 83–99, Jan. 2023. doi: 10.1002/cae.22572.

[43] A. K. Shukla, S. K. Pippal, and S. S. Chauhan, "An empirical evaluation of teaching-learning-based optimization, genetic algorithm and particle swarm optimization," *Int. J. Comput. Appl.*, vol. 45, no. 1, pp. 36–50, Jan. 2023. doi: 10.1080/1206212X.2019.1686562.

[44] J. Deng and L. Wang, "A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 32, pp. 121–131, Feb. 2017. doi: 10.1016/j.swevo.2016.06.002.