CAAI Transactions on Intelligence Technology

Special Issue: Advances in Bio-inspired Heuristics for Computing

TLBO with variable weights applied to shop scheduling problems

ISSN 2468-2322
Received on 13th March 2019
Revised on 22nd April 2019
Accepted on 27th May 2019
doi: 10.1049/trit.2018.1089
www.ietdl.org

Leonardo Ramos Rodrigues[1] ✉, *João Paulo Pordeus Gomes*[2]

[1]*Electronics Division, Institute of Aeronautics and Space, Praça Marechal Eduardo Gomes, 50, São José dos Campos 12228-904, Brazil*
[2]*Computer Science Department, Federal University of Ceará, Rua Campus do Pici, Sn, Fortaleza 60440-554, Brazil*
✉ *E-mail: leonardolrr2@fab.mil.br*

Abstract: The teaching–learning-based optimisation (TLBO) algorithm is a population-based metaheuristic inspired on the teaching–learning process observed in a classroom. It has been successfully used in a wide range of applications. In this study, the authors present a variant version of TLBO. In the proposed version, different weights are assigned to students during the student phase, with higher weights being assigned to students with better solutions. Three different approaches to assign weights are investigated. Numerical experiments with benchmark instances of the flow-shop and the job-shop scheduling problems are carried out to investigate the performance of the proposed approaches. They compare the proposed approaches with the original TLBO algorithm and with two variants of TLBOs proposed in the literature in terms of solution quality, convergence speed and simulation time. The results obtained by the application of a Friedman statistical test showed that the proposed approaches outperformed the original version of TLBO in terms of convergence, with no significant losses in the average makespan. The additional simulation time required by the proposed approaches is small. The best performance was achieved with the approach of assigning a fixed weight to half the students with the best solutions and assigning zero to other students.

1 Introduction

Heuristic optimisation algorithms are useful due to their ability to find good solutions to optimisation problems in an acceptable time [1]. Researchers have developed various heuristic optimisation techniques with different characteristics; many of them inspired by different phenomena observed in nature. Examples of heuristic techniques include particle swarm optimisation (PSO) [2], genetic algorithm (GA) [3] and ant colony optimisation [4].

Among all the heuristic optimisation methods proposed in the literature, the teaching–learning-based optimisation (TLBO) algorithm [5] is a promising one. As mentioned earlier, TLBO is a population-based algorithm that simulates the dynamics of teaching–learning processes. TLBO has been successfully used in a wide range of optimisation problems [6–10], with an additional advantage of a reduced number of parameters [11].

Several variants of TLBO have been developed to enhance the balance between the diversification and the intensification capabilities of the algorithm in order to improve its performance on complex optimisation problems. Incorporating elitism [11], variable population [12] and dynamic group strategy (DGS) [13] are among the successful variants of TLBO. Despite the good results achieved by these variants, there are still opportunities for further enhancements.

In this paper, we propose a new variant of the TLBO algorithm. In the proposed variant, different weights are assigned to students during the student phase, with higher weights being assigned to students with better solutions. The proposed approach is similar to the roulette wheel selection operator used in GAs [3], in which candidates with better fitness values have a higher probability of being chosen. We propose three different approaches to assign weights to students. We perform numerical experiments using benchmark instances of the classical flow-shop and job-shop scheduling problems, which are proven to be non-deterministic polynomial-time (NP)-hard combinatorial problems.

Both the flow-shop and the job-shop scheduling problems are combinatorial problems that appear in many practical situations. Also, previous researches have already shown that TLBO presents a competitive performance to solve these problems [6, 14]. So, the testbed used in this paper is appropriate to assess the performance of the proposed TLBO variants.

The remaining sections of this paper are organised as follows. Section 2 summarises the related papers on TLBO improvements. Section 3 describes the original version of TLBO and presents the shop scheduling problems considered in this paper. Section 4 introduces the proposed modified version of TLBO. Section 5 presents the results obtained with the proposed version of the TLBO algorithm in benchmark instances of the shop scheduling problems. Concluding remarks and suggestions for future research are presented in Section 6.

2 Related works

In the past few years, several papers proposing variants of TLBO have been published. In [11], Rao and Patel proposed an elitist TLBO (ETLBO). In the ETLBO, the best candidate solutions found in each iteration are preserved. In [15], Rao and Patel proposed the use of multiple teachers and the adaptation of the teaching factor aiming at improving the diversification and intensification capabilities of TLBO. In [16], Chen *et al.* incorporated local learning and self-learning methods in TLBO.

To decrease the computational cost and improve the global performance, Chen *et al.* [17] introduced the area copying operator of the producer–scrounger model in TLBO. In this model, the producer exploits new positions according to its ability along with a random angle and a maximal radius. The scroungers search around the producer. The convergence speed may be improved since producers and scroungers exploit the new positions when the diversity of the algorithm is lost.

In [12], *Chen et al.* proposed a modified TLBO with a variable population size (PS) to reduce the computational cost of TLBO. In the increasing phase, new individuals are generated according to a normal distribution with adaptive mean and variance. In the decreasing phase, individuals with high similarity are removed from the current population.

In [13], Zou *et al.* proposed a modified TLBO with a DGS (DGSTLBO). Different from the original TLBO algorithm, DGSTLBO enables each learner to learn from the mean of his corresponding group, rather than the mean of the whole class, during the teacher phase. The effectiveness of DGSTLBO was assessed with experiments using benchmark functions and showed promising results.

In [6], Baykasoğlu *et al.* investigated the performance of the basic version of TLBO in the solution of job-shop and flow-shop scheduling problems. They compared the TLBO performance in the solution of the flow-shop scheduling problem with the performance of other optimisation algorithms such as the novel PSO algorithm and the hybrid PSO algorithm, and showed that TLBO in its basic version provided a very close performance in comparison with known solutions in the literature. Similar results were observed for the job-shop scheduling problem, with TLBO in its basic version presenting a very close performance when compared with other optimisation methods such as the beam search algorithm [18] and the GRASP algorithm [19], among others.

Modified versions of TLBO to solve multi-objective optimisation problems have been developed [20–22]. The literature also contains hybrid methods that combine TLBO with other algorithms [23–25].

As can be noted, different strategies to reduce the computational cost or improve the capability of escaping from local minima have been incorporated in TLBO. Recent reviews on the application of TLBO in different classes of optimisation problems can be found in [26, 27]. Similarly, the variant proposed in this paper aims at reducing the computational cost of TLBO while preserving its optimisation performance. Moreover, the variable student weight strategies proposed in this paper can be incorporated in most of TLBO variants already proposed in a straightforward way.

3 Theoretical background

3.1 Teaching–learning-based optimisation

The TLBO algorithm is a population-based metaheuristic inspired by the dynamics of teaching–learning processes [5]. The algorithm is composed of two main phases, namely the teacher phase and the learner phase. The learner phase is often referred to as the student phase. The teacher phase simulates the learning process, in which students gain new knowledge from the teacher. The learner phase simulates the learning process, in which students learn through the interactions among themselves.

Each student produces a solution denoted by X. The solution X represents a candidate solution for the optimisation problem at hand. The solutions produced by the students are evaluated and a fitness value $f(X)$ that quantifies the quality of each solution is computed for all candidate solutions. In the flow-shop and the job-shop scheduling problems, each candidate solution X contains a sequence to execute the jobs in the available machines, and the fitness value of each solution represents the total amount of time needed to execute all the jobs, also called makespan.

The student with the best solution in each iteration of the algorithm is called the teacher. Fig. 1 illustrates the implementation of the TLBO algorithm [5]. The phases of the TLBO are described in more details in the next sections.

3.1.1 Teacher phase: During this phase, the algorithm mimics the process, in which each student gains knowledge from the teacher. Let n be the total number of students and M_i be the average solution among all the students. Also, let T_i be the teacher (i.e. the student with the best solution) at the ith iteration. The teacher will try to bring M_i to a value closer to X_{Ti}, which is the solution of the teacher. The difference between X_{Ti} and M_i, denoted by D_i, is defined as

$$D_i = r_i(X_{Ti} - T_F M_i) \tag{1}$$

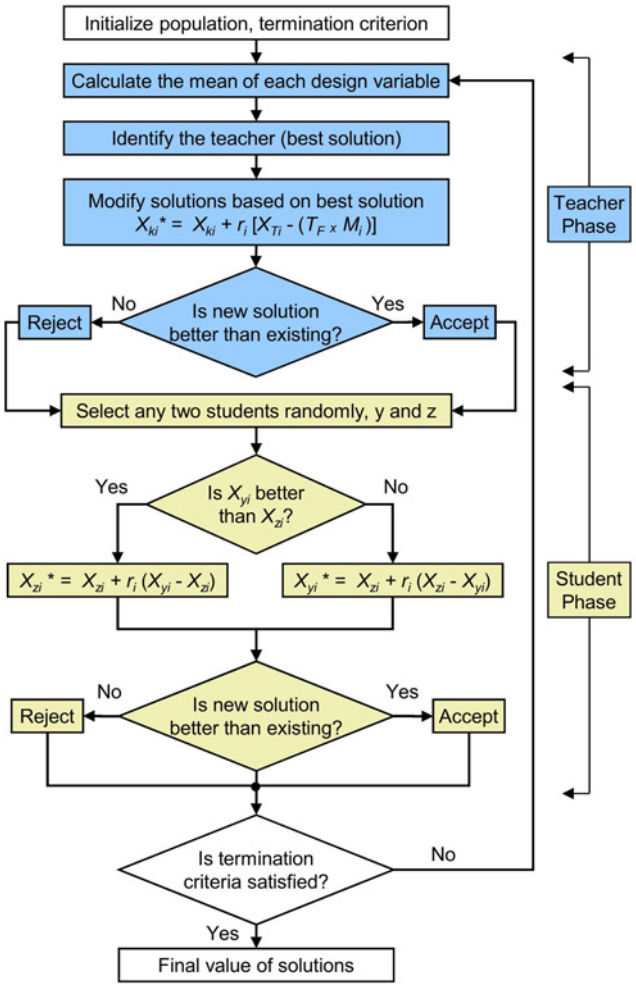

Fig. 1 *TLBO flowchart [5]*

where r_i is a random variable chosen from a uniform distribution in the range [0, 1] and T_F is the teaching factor that is randomly set to 1 or 2 in each iteration i, with equal probabilities, according to

$$T_F = \text{round}(1 + \text{rand}(0,\ 1)) \tag{2}$$

The difference D_i is used to update the current solution belonging to each student k in iteration i, denoted by X_{ki}, according to

$$X_{ki}^{\star} = X_{ki} + D_i \tag{3}$$

where $X_{ki}^{\star}$ is the updated value of X_{ki}.

Then, the fitness of each updated solution $X_{ki}^{\star}$, denoted by $f(X_{ki}^{\star})$, is computed. For each student, if $f(X_{ki}^{\star})$ is better than $f(X_{ki})$, then X_{ki} is replaced with $X_{ki}^{\star}$. Otherwise, $X_{ki}^{\star}$ is discarded and X_{ki} is kept for the next iteration.

3.1.2 Student/learner phase: During the learner phase, TLBO mimics the learning of the students through interactions among themselves. Let X_{yi} and X_{zi} be the solutions belonging to two students, namely y and z, at iteration i, respectively. If $f(X_{yi})$ is better than $f(X_{zi})$, X_{zi} is updated using (4). If $f(X_{zi}^{\star})$ is better than $f(X_{zi})$, X_{zi} is replaced with $X_{zi}^{\star}$. Otherwise, $X_{zi}^{\star}$ is discarded and X_{zi} is kept for the next iteration.

Similarly, if $f(X_{zi})$ is better than $f(X_{yi})$, X_{yi} is updated using (5). If $f(X_{yi}^{\star})$ is better than $f(X_{yi})$, X_{yi} is replaced with $X_{yi}^{\star}$. Otherwise, $X_{yi}^{\star}$ is discarded and X_{yi} is kept for the next iteration

$$X_{zi}^{\star} = X_{zi} + r_i(X_{yi} - X_{zi}) \tag{4}$$

$$X_{yi}^{\star} = X_{yi} + r_i(X_{zi} - X_{yi}) \tag{5}$$

Consider a population of n students. Also, consider a student k, with $1 \leq k \leq n$. Let $w(k)$ be the weight assigned to a student k that defines the relative probability of choosing a student k to interact during the learner phase. In the original TLBO, all the students have the same probability of being chosen, i.e. $w(k_1) = w(k_2)$, $\forall 1 \leq k_1 \leq n$ and $1 \leq k_2 \leq n$. The idea behind the proposed variant is to change the relative weights w based on the fitness value of each student k.

The stop conditions are verified after each iteration of the algorithm. In this paper, the algorithm ends after a predefined number of iterations.

3.2 Flow-shop scheduling problems

The flow-shop scheduling problem is a classical NP-hard combinatorial optimisation problem [28]. It has many practical applications in logistics, industrial and other fields [29]. The flow-shop scheduling problem has been widely studied for many decades [30].

In the flow-shop scheduling problem, the main goal is to find the sequence to execute all the jobs that result in the minimum makespan. The makespan is the amount of time required to complete all the jobs. A total of n jobs are considered, and each job consists of m operations. The flow shop is composed of m machines. The operations of all jobs must be executed following the same sequence, i.e. machines $m_1, \ldots, m_m$.

If a flow-shop scheduling problem has m machines and j jobs. For each possible sequence, the makespan can be computed as follows [6, 31]:

$$C(1,\ 1) = d(1,\ 1) \tag{6}$$

$$C(1,\ i) = C(1,\ i-1) + d(1,\ i) \tag{7}$$

$$C(r,\ 1) = C(r-1,\ 1) + d(r,\ 1) \tag{8}$$

$$C(r,\ i) = \max\left[C(r,\ i-1),\ C(r-1,\ i)\right] + d(r,\ i) \tag{9}$$

where $d(r, i)$ is the execution time for operation r $(1 \leq r \leq m)$ belonging to the job i $(1 \leq i \leq j)$ in the machine m_r and $C(r, i)$ is the time, in which operation r belonging to a job i in the machine m_r is finished.

As an example, consider that a flow shop contains four jobs and three machines. The execution time of each operation belonging to each job in each machine is defined as follows:

$$j_1 = [(1,\ 4),\ (2,\ 2),\ (3,\ 1)]$$

$$j_2 = [(1,\ 3),\ (2,\ 6),\ (3,\ 2)]$$

$$j_3 = [(1,\ 7),\ (2,\ 2),\ (3,\ 3)]$$

$$j_4 = [(1,\ 1),\ (2,\ 5),\ (3,\ 8)]$$

where a pair (m, d) indicates the machine m that executes the operation and the duration of the operation d.

Now, consider the candidate solution $P = [1, 4, 3, 2]$. Using (6)–(9), it is possible to compute the makespan for the permutation P. Fig. 2 shows the Gantt chart that represents the candidate solution P, which has a makespan of 24 time units.

A solution for the flow-shop scheduling problem is represented using a priority vectors approach [6]. Let $\boldsymbol{X}_k = \{x_k(1), \ldots, x_k(j)\}$ be a vector representing the solution of a student k, where each element $x_k(i)$ of the vector $\boldsymbol{X}_k$ is a real number that denotes the relative priority of the ith job.

To convert an $\boldsymbol{X}_k$ into a job sequence, the elements of the priority vector are sorted in decreasing order. Once the job sequence is found, the fitness value of the candidate solution represented by $\boldsymbol{X}_k$ is computed according to (6)–(9).

To illustrate how to map a priority vector $\boldsymbol{X}_k$ into a job permutation, consider a flow-shop problem containing six jobs and the priority vector $\boldsymbol{X}_k$ shown in Table 1. The highest value of $\boldsymbol{X}_k$ (8.407) is its fifth position, which means that job 5 will be executed before the others. Following a decreasing order of priority, the job permutation associated with student k will be $P = [5, 1, 6, 4, 3, 2]$.

3.3 Job-shop scheduling problem

The job-shop scheduling problem is one of the most studied and most complex problems in the scheduling research field [32, 33]. It is classified into the NP-complete group.

A job-shop scheduling problem is composed of a group of j jobs and m machines. Jobs are composed of a sequence of operations. Each operation has a predefined order and the machine where it must be executed [6]. Let $u(m, j)$ be the operation of a job j that requires a machine m. Also, let $d(m, j)$ be the duration of the operation $u(m, j)$. The objective of a job-shop scheduling problem is to find an execution sequence that minimises the amount of time required to execute all the operations.

Let $J = \{j_1, \ldots, j_j\}$ be the set of jobs, $M = \{m_1, \ldots, m_m\}$ be the set of machines and $U = \{u_0, u_1, \ldots, u_{j\times m}, u_{(j\times m)+1}\}$ be the set of all operations to be executed, where operations u_0 and $u_{(j\times m)+1}$ are the dummy initial and the dummy final operations, respectively. Also, let d_u and f_u be the execution time (duration) and the completion time of operation u, respectively.

The relations among operations are defined by two types of constraints: precedence constraints and machine constraints. Precedence constraints ensure that the execution of operation u will not start before the completion of all its predecessor operations, denoted by P_u. Machine constraints ensure that each machine executes only one operation at a time.

Let $A(t)$ denote the group of operations that are being executed at a time t. Also, let $r_{u,m}$ be a binary variable that assumes value 1 if operation u must be executed by a machine m and assumes value 0 otherwise. A candidate solution can be described by a vector $F = [f_0, \ldots, f_{(j\times m)+1}]$, where each element f_u of F is the completion time of operation u. Then, the job-shop scheduling problem can be formally defined as follows [6, 34]:

$$\text{minimise:} \quad f_{(j\times m)+1} \tag{10}$$

subjected to:

$$f_u - d_u \geq f_x, \quad \forall x \in P_u \tag{11}$$

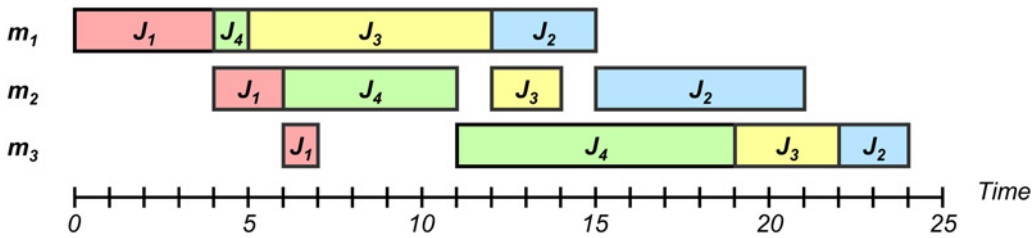

Fig. 2 *Gantt chart for the flow-shop example*

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 3, pp. 148–158

Table 1 Example of the conversion of a priority vector into a job sequence for the flow shop

i	1	2	3	4	5	6
$x_k(i)$	5.472	1.386	1.493	2.575	8.407	3.543

$$\sum_{u \in A(t)} r_{u,m} \leq 1, \quad m \in M;\ t \geq 0 \tag{12}$$

$$f_u \geq 0, \quad u = 1,\ \ldots,\ (j \times m) + 1 \tag{13}$$

where (10) is the objective function to be minimised, (11) represents the precedence relations between operations, (12) ensures that each machine will not execute multiple jobs simultaneously and (13) imposes non-negative finishing times.

A solution for the job-shop scheduling problem is represented by using a priority matrix approach [6]. Let $\boldsymbol{X}_k$ be an $m \times j$ real-number matrix containing the candidate solution belonging to the student k for a job-shop problem with m machines and j jobs, i.e.

$$\boldsymbol{X}_k = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mj} \end{bmatrix}$$

where each element x_{ri}, with $1 \leq r \leq m$ and $1 \leq i \leq j$, is the priority of the operation of job i that must be executed by a machine r.

The number of different candidate schedules for a job-shop scheduling problem with m machines and j jobs is $(j!)^m$. Since this number is very large even for a small number of machines and jobs, exploring the whole search space to guarantee the global minimum solution has a huge computational cost.

To reduce the search space, feasible schedules can be classified into four types [34]:

- *Inadmissible schedules*: Schedules, in which machines have excessive idle times. Schedules in this group can be improved by eliminating the excessive idle times. Excessive idle times are eliminating by shifting the operations.
- *Semi-active schedules*: These schedules are obtained when the operations are scheduling using the 'as early as possible' approach. Schedules in this group do not contain excessive idle times. However, they can still be improved by shifting some operations without causing delays in other operations.
- *Active schedules*: Schedules, in which no operation can be anticipated without causing delays in other operations (violating precedence restrictions). The global optimal is known to always be an active schedule. Thus, if the search space is reduced to the active schedules, the optimal solution can still be found.
- *Non-delay schedules*: Schedules, in which a machine is always executing an operation unless a precedence constraint is violated. Schedules in this group are always active schedules.

Fig. 3 illustrates the relations among the four schedule types. As mentioned earlier, the search space can be reduced to the subspace of active schedules without affecting the optimal solution [35]. In this paper, we use the Giffler & Thompson (G&T) algorithm to map a priority matrix $\boldsymbol{X}_k$ to an active schedule. A brief description of the G&T algorithm is given below [36].

3.3.1 G&T algorithm: The G&T algorithm is a recursive method to generate active schedules for the job-shop scheduling problem in a systematic way [36]. The notation adopted in the G&T algorithm is as follows:

- $u(m, j)$ is the operation belonging to job j that requires a machine m.

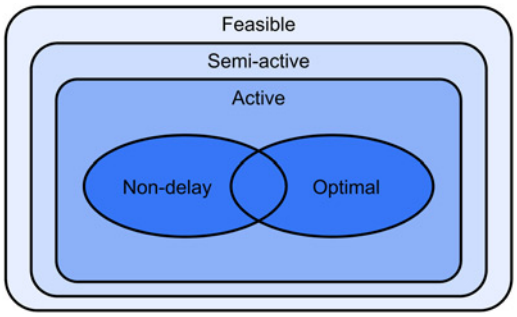

Fig. 3 *Relationship among schedule types*

1: **Step 1:** Initialize $S = \emptyset$ and X to the subset of all the operations without precedence restrictions.
2: **while** $X \neq \emptyset$ **do**
3: **Step 2:** Find operation $u(m,j)^\star \in X$ that has the earliest completion time $f(m,j)^\star$ and the corresponding machine $m^\star$ that will execute operation $u(m,j)^\star$.
4: **Step 3a:** Identify the subset of operations $Y \in X$ such that all operations in Y are executed on machine $m^\star$ and for which $e(m,j) < f^\star$.
5: **Step 3b:** Choose the operation $u(m,j)^{\star\star}$ from Y with the largest priority value.
6: **Step 3c:** Add $u(m,j)^{\star\star}$ to S.
7: **Step 3d:** Assign $e(m,j)^{\star\star}$ as the starting time of operation $u(m,j)^{\star\star}$.
8: **Step 4:** Delete $u(m,j)^{\star\star}$ from X and include its immediate successor, if any, in X.
9: **end while**

Fig. 4 *Algorithm 1: G&T algorithm*

- S is the partial schedule that contains all the operations already scheduled.
- X is the set of operations that are ready to be scheduled, i.e. the set of operations that have no precedence restrictions.
- $e(m, j)$ is the earliest time, in which operation $u(m, j)$ in X can start to be executed.
- $d(m, j)$ is the duration of the operation $u(m, j)$.
- $f(m, j)$ is the earliest time at which the execution of the operation $u(m, j)$ in X can finish, i.e. $f(m, j) = e(m, j) + d(m, j)$.

The implementation of the G&T algorithm is presented in Algorithm 1 (see Fig. 4) [6]:

To illustrate the use of the G&T algorithm, consider a job-shop scheduling problem composed of three jobs and three machines. The operations in each job are defined as follows [37]:

$$j_1 = [(2,\ 3),\ (1,\ 7),\ (3,\ 2)]$$

$$j_2 = [(1,\ 2),\ (3,\ 5),\ (2,\ 4)]$$

$$j_3 = [(2,\ 5),\ (3,\ 4),\ (1,\ 5)]$$

where a pair (m, d) describes the machine m that must execute the operation and the execution time d of the operation. All the operations belonging to each job must be executed in the presented order.

Also, consider the priority matrix $\boldsymbol{X}_k$ that represents the solution of a student k in TLBO

$$\boldsymbol{X}_k = \begin{bmatrix} 0.11 & 4.70 & 0.67 \\ 6.17 & 9.23 & 1.23 \\ 4.19 & 3.01 & 3.47 \end{bmatrix}$$

where each element x_{mj} is the relative priority value of the operation belonging to the jth job that is executed by the mth machine. The following lines describe all the steps to find a schedule associated with the priority matrix $\boldsymbol{X}_k$, according to Algorithm 1.

Initialisation:
Step 1: $S = \emptyset$ and $X = \{u(2,\ 1),\ u(1,\ 2),\ u(2,\ 3)\}$.
Iteration 1:
Step 2: $f(2,\ 1) = 3, f(1,\ 2) = 2$ and $f(2,\ 3) = 5$. Therefore, $f^{\star} = 2$ and $m^{\star} = 1$.
Step 3: $Y = \{u(1,\ 2)\}$. Operation $u(1,\ 2)$ is added to S, with $e(1,\ 2) = 0$.
Step 4: $u(1,\ 2)$ is deleted from X and $u(3,\ 2)$ is added to X. For the next iteration, $X = \{u(2,\ 1),\ u(3,\ 2),\ u(2,\ 3)\}$.
Iteration 2:
Step 2: $f(2,\ 1) = 3, f(3,\ 2) = 7$ and $f(2,\ 3) = 5$. Thus, $f^{\star} = 3$ and $m^{\star} = 2$.
Step 3: $Y = \{u(2,\ 1),\ u(2,\ 3\}$. The operation $u(2,\ 1)$ has the highest priority (6.17) and is added to S, with $e(2,\ 1) = 0$.
Step 4: $u(2,\ 1)$ is deleted from X and $u(1,\ 1)$ is added to X. For the next iteration, $X = \{u(1,\ 1),\ u(3,\ 2),\ u(2,\ 3)\}$.
Iteration 3:
Step 2: $f(1,\ 1) = 10$, $f(3,\ 2) = 7$ and $f(2,\ 3) = 8$. Thus, $f^{\star} = 7$ and $m^{\star} = 3$.
Step 3: $Y = \{u(3,\ 2)\}$. Operation $u(3,\ 2)$ is added to S, with $e(3,\ 2) = 2$.
Step 4: $u(3,\ 2)$ is deleted from X and $u(2,\ 2)$ is added to X. For the next iteration, $X = \{u(1,\ 1),\ u(2,\ 2),\ u(2,\ 3)\}$.
Iteration 4:
Step 2: $f(1,\ 1) = 10, f(2,\ 2) = 11$ and $f(2,\ 3) = 8$. Thus, $f^{\star} = 8$ and $m^{\star} = 2$.
Step 3: $Y = \{u(2,\ 2),\ (2,\ 3)\}$. The operation $u(2,\ 2)$ has the highest priority (9.23) and is added to S, with $e(2,\ 2) = 7$.
Step 4: $u(2,\ 2)$ is deleted from X. No operation is added to X because $u(2,\ 2)$ is the last operation of the job j_2. For the next iteration, $X = \{u(1,\ 1),\ u(2,\ 3)\}$.
Iteration 5:
Step 2: $f(1,\ 1) = 10$ and $f(2,\ 3) = 16$. Thus, $f^{\star} = 10$ and $m^{\star} = 1$.
Step 3: $Y = \{u(1,\ 1)\}$. Operation $u(1,\ 1)$ is added to S, with $e(1,\ 1) = 3$.
Step 4: $u(1,\ 1)$ is deleted from X and $u(3,\ 1)$ is added to X. For the next iteration, $X = \{u(3,\ 1),\ u(2,\ 3)\}$.
Iteration 6:
Step 2: $f(3,\ 1) = 12$ and $f(2,\ 3) = 16$. Thus, $f^{\star} = 12$ and $m^{\star} = 3$.
Step 3: $Y = \{u(3,\ 1)\}$. Operation $u(3,\ 1)$ is added to S, with $e(3,\ 1) = 10$.
Step 4: $u(3,\ 1)$ is deleted from X. No operation is added to X because $u(3,\ 1)$ is the last operation of the job j_1. For the next iteration, $X = \{u(2,\ 3)\}$.
Iterations 7, 8 and 9:
Now, all operations which have not been scheduled belong to the job j_3. Therefore, in iterations 7, 8 and 9 the operations of the job j_3 will be scheduled following the sequence of job j_3, i.e. $u(2,\ 3)$, $u(3,\ 3)$ and $u(1,\ 3)$.

Fig. 5 shows the Gantt chart for the solution, which has a makespan of 25.

4 Proposed method

This section describes the proposed modification in the TLBO algorithm. In the proposed variant, different weights are assigned to each student during the student phase, with higher weights being assigned to students with better solutions. The idea behind the proposed method is that good students are more likely to be invited to participate in study groups.

A similar concept was proposed in [3] for the selection operator of GAs. In GAs, the selection operator chooses two parents from the current generation to reproduce and generate a new child with the help of crossover and mutation operators [38]. In the selection operator proposed in [3], called roulette wheel, the probability of an individual chromosome being selected is directly related to its fitness.

Consider a population of n students. Let q be the rank index of each student, which indicates the relative position of the solution associated with that student. The rank index of the best student is 1. Similarly, the rank index of the worst student is n.

Let $w(q)$ be the weight assigned to the student whose rank index is q, with $1 \leq q \leq n$. In this paper, three different approaches are investigated to assign the weights to students. The proposed approaches are as follows:

- *Approach 1*: In this first approach, a weight of 1 is assigned to half the students with the best solutions and a weight of zero is assigned to the other students, according to

$$w(q) = 1; \quad 1 \leq q \leq n/2 \tag{14}$$

$$w(q) = 0; \quad q > n/2 \tag{15}$$

- *Approach 2*: In this second approach, the weight assigned to each student is based on the rank index q, as defined in (16). In other words, a weight of 1 is assigned to the worst student, a weight of 2 is assigned to the second worst student and so on. This procedure continues until the best student, who receives a weight of n

$$w(q) = n - q + 1; \quad 1 \leq q \leq n \tag{16}$$

- *Approach 3*: In this third approach, the weight assigned to half the students with the worst solutions follows the same rule used in approach 2. A fixed weight of $n/2$ is assigned to the other students, as presented in the equations below:

$$w(q) = n/2; \quad 1 \leq q \leq n/2 \tag{17}$$

$$w(q) = n - q + 1; \quad q > n/2 \tag{18}$$

Fig. 6*a* shows a graphical representation of the weights assigned to each student in the original TLBO. The weights assigned to each student in proposed approaches 1, 2 and 3 are illustrated in Figs. 6*b–d*, respectively.

Let $p(q)$ be the probability of student whose rank index is q being selected during the student phase. The computation of $p(q)$ is made according to the equation below:

$$p(q) = \frac{w(q)}{\sum_{v=1}^{n} w(v)} \tag{19}$$

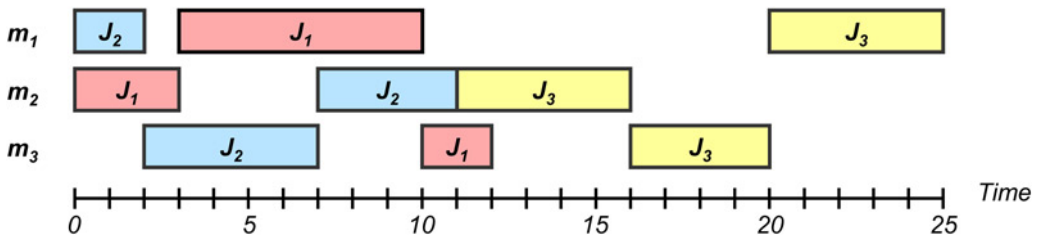

Fig. 5 *Gantt chart for the job shop*

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 3, pp. 148–158

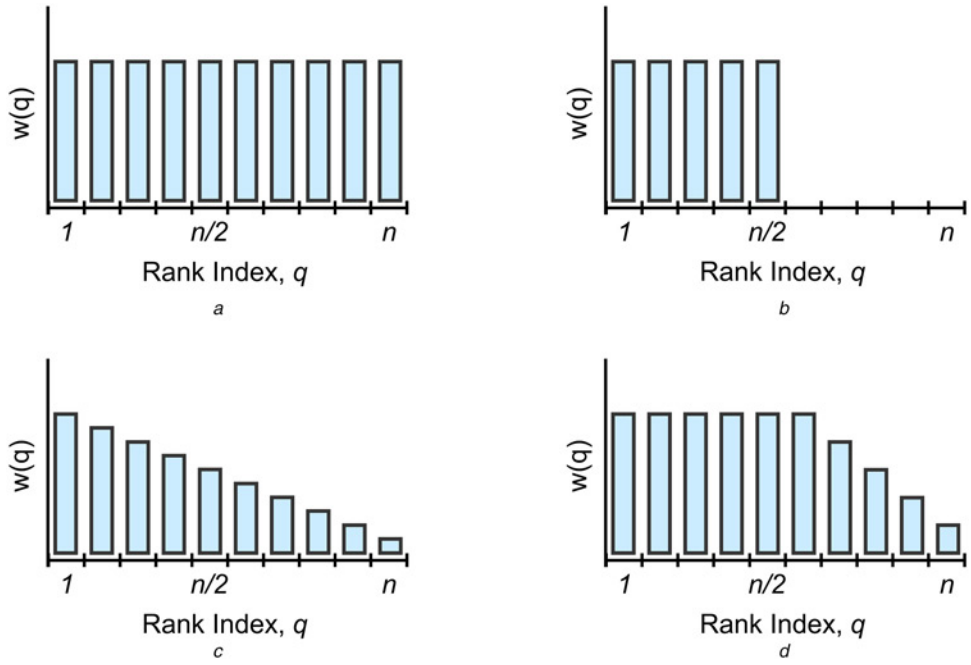

Fig. 6 *Graphical representation of different approaches to assign weights to each student during the student phase of TLBO*

a Original TLBO,
b Proposed approach 1,
c Proposed approach 2,
d Proposed approach 3

5 Numerical experiments

In this section, we present the results observed during the numerical experiments carried out to investigate the performance of the proposed approaches to assign weights to students in benchmark instances of the flow-shop and the job-shop scheduling problems.

About 20 benchmark instances of the flow shop and 20 benchmark instances of the job-shop scheduling problem were used in the numerical experiments. Tables 2 and 3 present the number of jobs and machines in each instance of the flow-shop and the job-shop scheduling problems, respectively.

Table 2 Flow-shop benchmark instances

Instance	Jobs	Machines	Instance	Jobs	Machines
car1	11	5	reC17	20	15
car6	8	9	reC19	30	10
car7	7	7	reC23	30	10
hel1	100	10	reC25	30	15
hel2	20	10	reC29	30	15
reC03	20	5	reC33	50	10
reC05	20	5	reC35	50	10
reC09	20	10	reC37	75	20
reC11	20	10	reC39	75	20
reC15	20	15	reC41	75	20

Table 3 Job-shop benchmark instances

Instance	Jobs	Machines	Instance	Jobs	Machines
abz5	10	10	la31	30	10
abz7	20	15	la35	30	10
ft06	6	6	la36	15	15
ft20	20	5	orb05	10	10
la06	15	5	orb09	10	10
la11	20	5	swv01	20	10
la12	20	5	swv04	20	10
la16	10	10	swv05	20	10
la22	15	10	swv16	50	10
la26	20	10	yn1	20	20

5.1 Parameter settings

The performance of metaheuristic algorithms is highly dependent on the choice of parameter values. For the TLBO algorithm, the parameters to be defined are the PS and the maximum number of generations (GNs). In the absence of a mathematical model to determine the optimal values for these parameters, they are commonly determined empirically. In this paper, a design of experiments (DOE) approach was used to find out good settings for the parameters. The DOE approach is a method used to investigate and evaluate the effect of multiple factors on a process [39].

For the flow-shop scheduling problem, instance reC23 was used in the DOE method because it has an intermediate complexity level. Five different candidate values were considered for GNs 200, 300, 400, 500 and 600. Five different values were also considered for PSs 30, 60, 90, 120 and 150. A full factorial experimental layout was used. For each combination, 30 runs were carried out.

To identify the parameters that have a significant effect, a statistical analysis of variance (ANOVA) was carried out. The results are presented in Table 4. All the main effects are considered significant at the level of 5%. In Table 4, df is the number of degrees of freedom, SS is the sum of squares, MS is the mean square, F is the F-test statistics and p is the probability value used to test the null hypothesis, in which a parameter effect is not significant.

Tables 5 and 6 show the average makespan computed for each value of GN and PS, respectively. On the basis of these results, the final values for GN and PS for the flow-shop scheduling problem were chosen as 300 and 90, respectively. These values

Table 4 ANOVA results for flow-shop instance reC23

Source	SS	df	MS	F	p
GN	231.1	4	57.8	3.19	0.042
PS	3905.4	4	976.3	53.93	0.000
Error	289.7	16	18.1	—	—
Total	4426.1	24	—	—	—

will be used in all the experiments conducted for the flow-shop scheduling problem.

For the job-shop scheduling problem, instance la26 was used in the DOE method because it has an intermediate complexity level. The same values of GN and PS used for the flow-shop scheduling problem were used here. A full factorial experimental layout was conducted. For each combination, 30 runs were carried out.

Table 5 Average makespan with different values of GN for flow-shop instance reC23

GN	200	300	400	500	600
makespan	2110.4	2107.7	2115.8	2114.0	2109.3

Table 6 Average makespan with different values of PS for flow-shop instance reC23

PS	30	60	90	120	150
makespan	2130.0	2121.1	2097.2	2108.9	2099.9

Table 7 ANOVA results for job-shop instance la26

Source	*SS*	*df*	*MS*	*F*	*p*
GN	2645.8	4	661.4	3.44	0.033
PS	3683.4	4	920.9	4.78	0.010
error	3079.5	16	192.5	—	—
Total	9408.7	24	—	—	—

Table 8 Average makespan with different values of GN for job-shop instance la26

GN	200	300	400	500	600
makespan	1480.0	1464.6	1448.5	1462.9	1457.8

Table 9 Average makespan with different values of PS for job-shop instance la26

PS	30	60	90	120	150
makespan	1481.6	1466.9	1455.0	1445.5	1464.9

A statistical ANOVA was also carried out. The results are presented in Table 7. All the main effects are considered significant at the level of 5%.

Tables 8 and 9 show the mean makespan computed for each value of GN and PS, respectively. On the basis of these results, the final values for GN and PS for the job-shop scheduling problem were chosen as 400 and 120, respectively. These values will be used in all the experiments conducted for the job-shop scheduling problem.

5.2 Simulation results

5.2.1 Convergence speed: In this first experiment, our goal is to investigate the performance of the proposed approaches in terms of convergence speed. The proposed approaches are compared with the original TLBO. Two variants of TLBO proposed in the literature are also considered in the experiments: the ETLBO [11] and the variable population scheme TLBO (VPTLBO) [12]. For each benchmark instance, we carried out 30 runs of each algorithm. For each run, we computed the GNs that the algorithm ran until finding the final solution. Tables 10 and 11 show the average GNs obtained for the benchmark instances of the flow-shop and the job-shop scheduling problems, respectively.

It can be noted from Tables 10 and 11 that the original TLBO was outperformed by at least one of the proposed approaches in all benchmark instances considered in this paper. Also, in 12 out of the 20 instances of the flow shop, and in 16 out of the 20 instances of the job shop, the original TLBO was outperformed by all the proposed approaches. VPTLBO was outperformed by at least one of the proposed approaches in 19 out of the 20 flow-shop instances. However, VPTLBO presented the best result in four instances of the job-shop scheduling problem.

Another interesting result observed in this experiment is the superior performance of approach 1, which presented the best performance among all methods in 10 and 11 instances of the flow shop and the job shop, respectively. Also, the proposed approach 1 outperformed the original TLBO in all benchmark instances considered in this paper.

We performed a statistical test to verify whether the superior performance of the proposed approach 1 is statistically significant. For this purpose, we used the Friedman test [40, 41]. The Friedman test is a non-parametric procedure that can be used to compare multiple algorithms and detect significant differences between their results [42].

The null hypothesis H_0 for the Friedman test is that there are no significant differences in the performances of the methods. The first step to compute the Friedman statistics is to convert the original data into ranks. For each scheduling problem (flow shop

Table 10 Average GNs to find the final solution for the flow-shop problems

Instance	Original TLBO	Approach 1	Approach 2	Approach 3	ETLBO	VPTLBO
car1	3.9 ✗	2.8 ✓	3.6 ✗	2.9 ✗	6.5 ✗	5.1 ✗
car6	43.1 ✗	41.6 ✗	25.1 ✓	33.3 ✗	117.1 ✗	162.9 ✗
car7	13.3 ✗	12.8 ✓	13.9 ✗	13.9 ✗	13.1 ✗	20.9 ✗
hel1	123.0 ✗	106.5 ✓	111.9 ✗	126.0 ✗	199.6 ✗	127.6 ✗
hel2	96.1 ✗	76.6 ✗	87.2 ✗	86.1 ✗	61.7 ✓	82.7 ✗
reC03	61.5 ✗	58.5 ✓	57.5 ✗	79.0 ✗	60.7 ✗	64.5 ✗
reC05	63.5 ✗	✗.2 ✗	56.7 ✗	40.9 ✓	101.9 ✗	133.4 ✗
reC09	73.3 ✗	61.9 ✓	65.5 ✗	65.8 ✗	116.2 ✗	102.3 ✗
reC11	100.1 ✗	72.3 ✗	94.8 ✗	99.2 ✗	62.1 ✓	164.7 ✗
reC15	87.5 ✗	85.7 ✗	88.4 ✗	91.9 ✗	74.4 ✓	79.8 ✗
reC17	101.6 ✗	79.4 ✓	101.1 ✗	85.2 ✗	84.7 ✗	150.7 ✗
reC19	109.2 ✗	86.5 ✓	126.6 ✗	110.0 ✗	97.8 ✗	125.3 ✗
reC23	99.9 ✗	82.8 ✗	105.1 ✗	102.7 ✗	68.8 ✓	87.7 ✗
reC25	140.6 ✗	93.1 ✗	127.6 ✗	112.3 ✗	85.2 ✓	168.5 ✗
reC29	124.2 ✗	96.0 ✓	97.2 ✗	104.8 ✗	108.4 ✗	171.7 ✗
reC33	84.3 ✗	69.4 ✓	89.7 ✗	95.1 ✗	105.9 ✗	113.6 ✗
reC35	84.4 ✗	60.7 ✓	75.8 ✗	80.9 ✗	99.6 ✗	166.6 ✗
reC37	224.5 ✗	173.8 ✗	207.4 ✗	210.0 ✗	161.0 ✓	267.6 ✗
reC39	196.5 ✗	153.0 ✗	188.5 ✗	180.0 ✗	131.4 ✓	201.9 ✗
reC41	197.3 ✗	173.0 ✗	219.8 ✗	217.5 ✗	130.0 ✓	227.9 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 3, pp. 148–158

Table 11 Average GNs to find the final solution for the job-shop problems

Instance	Original TLBO	Approach 1	Approach 2	Approach 3	ETLBO	VPTLBO
abz5	197.4 ✗	113.4 ✓	153.6 ✗	150.8 ✗	150.6 ✗	145.7 ✗
abz7	238.7 ✗	235.8 ✓	240.9 ✗	253.2 ✗	275.3 ✗	292.9 ✗
ft06	87.1 ✗	57.5 ✗	23.3 ✓	50.2 ✗	50.6 ✗	65.5 ✗
ft20	244.1 ✗	240.3 ✗	304.0 ✗	219.1 ✗	185.7 ✗	181.7 ✓
la06	51.7 ✗	33.3 ✓	38.8 ✗	61.6 ✗	128.0 ✗	79.6 ✗
la11	106.6 ✗	83.2 ✓	120.6 ✗	97.1 ✗	210.8 ✗	97.2 ✗
la12	146.5 ✗	138.8 ✗	144.9 ✗	105.1 ✗	369.0 ✗	100.6 ✓
la16	160.0 ✗	94.8 ✗	140.4 ✗	153.6 ✗	77.7 ✓	100.1 ✗
la22	317.4 ✗	196.1 ✗	282.6 ✗	280.0 ✗	105.4 ✓	189.1 ✗
la26	307.1 ✗	261.7 ✗	218.9 ✗	269.3 ✗	197.9 ✗	185.8 ✓
la31	307.9 ✗	240.4 ✓	265.7 ✗	266.0 ✗	290.4 ✗	294.2 ✗
la35	300.4 ✗	248.2 ✗	254.5 ✗	270.8 ✗	175.3 ✓	183.2 ✗
la36	307.3 ✗	260.2 ✗	288.0 ✗	200.3 ✓	290.4 ✗	287.6 ✗
orb05	169.3 ✗	118.2 ✓	148.0 ✗	123.0 ✗	137.7 ✗	176.8 ✗
orb09	209.1 ✗	152.6 ✓	171.4 ✗	178.7 ✗	163.0 ✗	181.6 ✗
swv01	308.4 ✗	261.3 ✓	278.6 ✗	280.8 ✗	322.9 ✗	286.1 ✗
swv04	310.1 ✗	2*x*.0 ✗	264.2 ✗	275.8 ✗	260.2 ✗	240.0 ✓
swv05	306.0 ✗	243.4 ✓	247.0 ✗	248.9 ✗	252.6 ✗	294.4 ✗
swv16	318.6 ✗	264.1 ✓	270.1 ✗	268.7 ✗	275.2 ✗	293.3 ✗
yn1	262.8 ✗	201.7 ✓	229.1 ✗	242.2 ✗	252.7 ✗	294.6 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

or job shop), for each benchmark instance i, with $(1 \leq i \leq n)$, the results obtained from the algorithms are ranked from 1 (best result) to k (worst result). In case of ties, averaged ranks are used for tied algorithms. Then, for each algorithm j, with $(1 \leq j \leq k)$, the final algorithm rank R_j is defined as

$$R_j = \sum_{i=1}^{n} r_{ij} \tag{20}$$

where r_{ij} is the rank obtained in benchmark instance i with the algorithm j.

Table 12 Friedman test result for algorithm convergence speed

Statistics	Value
F_C	11.0705
F (flow shop)	40.9643
F (job shop)	33.4571

The Friedman statistics is defined as

$$F = \frac{12n}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{21}$$

where n is the number of benchmark instances (20 in our experiments) and k is the number of algorithms (6 in our experiments).

The Friedman statistics F must be compared with the critical value F_C, which is calculated using a χ^2 distribution with $k-1$ degrees of freedom and a significance level α. In this paper, we assumed $\alpha = 0.05$. To reject the null hypothesis H_0, the condition $F > F_C$ must be satisfied. Table 12 shows the results obtained for the statistical test.

On the basis of the statistical test result, we can reject the null hypothesis H_0 and consider that the proposed weight assignment approaches increased the convergence capability of TLBO.

This result was expected because of the higher probability of selecting good students introduced by the proposed approaches strengthens the intensification capability of the method. One

Table 13 Flow-shop simulation results

Instance	BKS	Original TLBO		Approach 1		Approach 2		Approach 3		ETLBO		VPTLBO	
		Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best
car1	7038	7038.0 ✓	7038 ✓	7040.3 ✗	7038 ✓	7038.1 ✗	7038 ✓	7038.0 ✓	7038 ✓	7038.0 ✓	7038 ✓	7038.0 ✓	7038 ✓
car6	8505	8539.1 ✗	8505 ✓	8532.6 ✗	8505 ✓	8530.6 ✗	8505 ✓	8534.9 ✗	8505 ✓	8528.2 ✓	8505 ✓	8538.7 ✗	8505 ✓
car7	6590	6593.4 ✗	6590 ✓	6590.0 ✓	6590 ✓	6595.6 ✗	6590 ✓	6591.1 ✗	6590 ✓	6591.7 ✗	6590 ✓	6591.3 ✗	6590 ✓
hel1	516	528.9 ✗	519 ✓	528.2 ✗	521 ✗	528.1 ✗	520 ✗	528.0 ✗	520 ✗	536.5 ✗	519 ✓	522.9 ✓	520 ✗
hel2	136	139.1 ✗	136 ✓	139.1 ✗	137 ✗	139.3 ✗	137 ✗	138.6 ✓	136 ✓	139.0 ✗	137 ✗	140.1 ✗	137 ✗
reC03	1109	1123.3 ✓	1111 ✗	1125.2 ✗	1110 ✗	1125.6 ✗	1109 ✓	1124.7 ✗	1111 ✗	1123.9 ✗	1111 ✗	1127.9 ✗	1109 ✓
reC05	1242	1255.8 ✗	1245 ✓	1257.4 ✗	1245 ✓	1257.4 ✗	1245 ✓	1260.2 ✗	1245 ✓	1254.4 ✓	1245 ✓	1256.8 ✗	1245 ✓
reC09	1537	1580.4 ✗	1538 ✗	1578.3 ✓	1538 ✗	1580.3 ✗	1546 ✗	1579.6 ✗	1537 ✓	1579.5 ✗	1538 ✗	1580.4 ✗	1538 ✗
reC11	1431	1474.2 ✓	1431 ✓	1488.8 ✗	1441 ✗	1484.3 ✗	1438 ✗	1481.1 ✗	1445 ✗	1508.3 ✗	1441 ✗	1485.3 ✗	1431 ✓
reC15	1950	2017.7 ✗	1967 ✗	2008.1 ✓	1964 ✓	2010.8 ✗	1966 ✗	2010.2 ✗	1964 ✓	2013.4 ✗	1978 ✗	2015.0 ✗	1967 ✗
reC17	1902	1973.1 ✓	1924 ✓	1979.7 ✗	1929 ✗	1987.9 ✗	1937 ✗	1976.1 ✗	1933 ✗	1982.9 ✗	1969 ✗	1981.4 ✗	1942 ✗
reC19	2093	2208.6 ✗	2142 ✗	2208.2 ✗	2158 ✗	2201.0 ✗	2137 ✓	2196.6 ✓	2141 ✗	2201.9 ✗	2156 ✗	2206.9 ✗	2145 ✗
reC23	2011	2110.1 ✗	2067 ✗	2107.5 ✗	2062 ✗	2113.8 ✗	2069 ✗	2104.6 ✗	2046 ✓	2108.6 ✗	2046 ✓	2101.0 ✓	2060 ✗
reC25	2513	2652.8 ✗	2583 ✗	2642.1 ✗	2591 ✗	2649.3 ✗	2602 ✗	2653.4 ✗	2577 ✓	2638.9 ✓	2608 ✗	2645.1 ✗	2601 ✗
reC29	2287	2444.1 ✗	2369 ✗	2456.7 ✗	2371 ✗	2442.4 ✗	2361 ✓	2445.9 ✗	2378 ✗	2449.7 ✗	2361 ✓	2442.0 ✓	2377 ✗
reC33	3114	3245.3 ✗	3173 ✗	3253.6 ✗	3187 ✗	3249.8 ✗	3164 ✓	3245.1 ✓	3171 ✗	3245.5 ✗	3173 ✗	3258.4 ✗	3173 ✗
reC35	3277	3374.9 ✗	3292 ✗	3366.9 ✓	3288 ✓	3378.7 ✗	3288 ✓	3369.0 ✗	3288 ✓	3374.5 ✗	3292 ✗	3370.7 ✗	3288 ✓
reC37	4951	5408.8 ✗	5267 ✓	5400.1 ✗	5320 ✗	5400.8 ✗	5306 ✗	5406.1 ✗	5309 ✗	5398.6 ✓	5320 ✗	5408.3 ✗	5334 ✗
reC39	5161	5503.3 ✗	5387 ✗	5491.0 ✗	5377 ✓	5486.6 ✓	5392 ✗	5488.4 ✗	5382 ✗	5488.3 ✗	5397 ✗	5489.0 ✗	5392 ✗
reC41	5087	5464.7 ✗	5342 ✗	5437.0 ✗	5352 ✗	5428.3 ✓	5342 ✗	5436.9 ✗	5326 ✓	5430.2 ✗	5342 ✗	5458.0 ✗	5356 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

Table 14 Job-shop simulation results

Instance	BKS	Original TLBO		Approach 1		Approach 2		Approach 3		ETLBO		VPTLBO	
		Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best
abz5	1234	1333.6 ✗	1285 ✓	1338.1 ✗	1314 ✗	1343.8 ✗	1307 ✗	1353.9 ✗	1318 ✗	1333.2 ✓	1307 ✗	1348.0 ✗	1320 ✗
abz7	656	826.8 ✗	810 ✗	825.6 ✗	809 ✓	823.1 ✓	820 ✗	827.6 ✗	811 ✗	827.5 ✗	810 ✗	825.9 ✗	811 ✗
ft06	55	56.0 ✗	55 ✓	55.6 ✗	55 ✓	55.9 ✗	55 ✓	55.3 ✓	55 ✓	56.2 ✗	55 ✓	55.7 ✗	55 ✓
ft20	1165	1224.5 ✗	1192 ✓	1222.4 ✓	1201 ✗	1239.0 ✗	1216 ✗	1228.6 ✗	1214 ✗	1226.0 ✗	1212 ✗	1222.6 ✗	1214 ✗
la06	926	926.0 ✓	926 ✓	926.0 ✓	926 ✓	926.0 ✓	926 ✓	926.0 ✓	926 ✓	926.0 ✓	926 ✓	926.0 ✓	926 ✓
la11	1222	1222.0 ✓	1222 ✓	1224.0 ✗	1222 ✓	1223.6 ✗	1222 ✓	1222.5 ✗	1222 ✓	1227.2 ✗	1222 ✓	1224.3 ✗	1222 ✓
la12	1039	1040.3 ✗	1039 ✓	1039.5 ✓	1039 ✓	1039.6 ✗	1039 ✓	1040.2 ✗	1039 ✓	1041.5 ✗	1039 ✓	1040.2 ✗	1039 ✓
la16	945	1006.6 ✗	991 ✗	1007.3 ✗	981 ✗	1004.4 ✗	995 ✗	1013.5 ✗	968 ✓	1007.1 ✗	975 ✗	1003.9 ✓	991 ✗
la22	927	1069.9 ✗	1032 ✓	1074.6 ✗	1054 ✗	1091.1 ✗	1033 ✗	1096.5 ✗	1063 ✗	1068.1 ✓	1046 ✗	1086.4 ✗	1058 ✗
la26	1218	1459.9 ✗	1435 ✗	1462.6 ✗	1433 ✗	1447.4 ✓	1408 ✓	1460.6 ✗	1431 ✗	1503.4 ✗	1433 ✗	1496.1 ✗	1435 ✗
la31	1784	1942.2 ✗	1897✗	1938.3 ✓	1911✗	1941.0✗	1893 ✓	1945.6✗	1901✗	1945.9✗	1893 ✓	1968.6✗	1916✗
la35	1888	2027.3✗	1990 ✗	2026.3 ✓	1986 ✗	2046.6 ✗	1981 ✗	2047.0 ✗	1974 ✓	2058.3 ✗	1985 ✗	2027.6 ✗	1992 ✗
la36	1268	1475.7 ✗	1417 ✓	1474.1 ✓	1428 ✗	1477.6 ✗	1457 ✗	1484.1 ✗	1444 ✗	1475.3 ✗	1461 ✗	1475.8 ✗	1457 ✗
orb05	887	954.9 ✗	908 ✓	964.3 ✗	931 ✗	946.9 ✓	929 ✗	953.9 ✗	927 ✗	970.4 ✗	932 ✗	963.1 ✗	931 ✗
orb09	934	1005.8 ✗	979 ✗	997.3 ✗	975 ✗	996.8 ✗	971 ✗	994.2 ✓	962 ✗	995.7 ✗	958 ✓	1009.1 ✗	962 ✗
swv01	1407	1668.0 ✗	1595 ✓	1663.6 ✗	1621 ✗	1685.8 ✗	1644 ✗	1685.8 ✗	1651 ✗	1673.7 ✗	1631 ✗	1662.9 ✓	1644 ✗
swv04	1470	1743.7 ✗	1663 ✗	1738.5 ✗	1661 ✓	1753.6 ✗	1679 ✗	1746.3 ✗	1696 ✗	1725.6 ✓	1701 ✗	1745.6 ✗	1663 ✗
swv05	1424	1712.9 ✗	1679 ✗	1709.1 ✓	1639 ✓	1717.0 ✗	1669 ✗	1726.8 ✗	1654 ✗	1710.0 ✗	1673 ✗	1737.9 ✗	1682 ✗
swv16	2924	2977.7 ✗	2939 ✓	2967.4 ✗	2940 ✗	2999.0 ✗	2946 ✗	2972.4 ✗	2950 ✗	2963.2 ✓	2953 ✗	2969.5 ✗	2946 ✗
yn1	884	1115.3 ✗	1078 ✗	1110.9 ✗	1087 ✗	1105.8 ✓	1065 ✓	1107.5 ✗	1074 ✗	1127.1 ✗	1097 ✗	1113.2 ✗	1074 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

possible disadvantage of the proposed approaches is that the algorithm may lose its capability to escape from local minima. In the next experiment, we compare the performance of the algorithms in terms of makespan.

5.2.2 Solution quality: In this second experiment, our goal is to compare the performance of the proposed approaches with the performances of the original TLBO, ETLBO and VPTLBO in terms of solution quality, i.e. in terms of makespan. For each benchmark instance, we carried out 30 runs of each algorithm. Tables 13 and 14 show the average and the best makespan provided by each algorithm for each benchmark instance of the flow-shop and the job-shop scheduling problems, respectively. In these tables, BKS is the best known solution for each instance, presented for reference purposes.

It can be noted from Tables 13 and 14 that the best performances in terms of average makespan and best makespan for both the flow-shop and the job-shop scheduling problems are distributed among the algorithms considered in the experiments.

Again, we used the Friedman test to analyse the results. As we observed an increase in convergence speed of the proposed approaches, our goal here is to investigate whether this increase in convergence speed is followed by a decrease in the quality of final solutions.

We consider the same significance level $\alpha = 0.05$. We computed the Friedman statistics for both the average and the best (lowest) makespans found by each algorithm. Table 15 shows the results.

On the basis of the statistical test result, we cannot reject the null hypothesis H_0 for this second experiment. Then, we can conclude that the proposed approaches provided a significant increase in the convergence speed of the method, without penalising the quality of solutions.

Table 15 Friedman test result for quality of solutions

Statistics	Average value	Best value
F_C	11.0705	11.0705
F (flow shop)	3.0350	2.2643
F (job shop)	9.8571	2.0357

Table 16 Simulation time, in seconds, for the flow-shop problems

Instance	Original TLBO	Approach 1	Approach 2	Approach 3	ETLBO	VPTLBO
car1	12.50 ✗	12.48 ✓	16.21 ✗	13.37 ✗	12.89 ✗	15.40 ✗
car6	13.59 ✓	15.16 ✗	24.57 ✗	16.09 ✗	15.41 ✗	17.94 ✗
car7	12.79 ✓	14.45 ✗	17.96 ✗	14.11 ✗	15.31 ✗	17.74 ✗
hel1	47.70 ✗	50.07 ✗	55.49 ✗	52.00 ✗	44.69 ✓	54.00 ✗
hel2	18.99 ✓	20.58 ✗	24.31 ✗	21.80 ✗	21.18 ✗	23.60 ✗
reC03	15.91 ✓	16.60 ✗	20.68 ✗	17.89 ✗	17.38 ✗	19.81 ✗
reC05	16.82 ✗	15.75 ✗	17.37 ✗	17.52 ✗	14.32 ✓	19.24 ✗
reC09	18.07 ✓	19.11 ✗	22.04 ✗	21.36 ✗	18.76 ✗	22.22 ✗
reC11	18.47 ✓	21.97 ✗	24.61 ✗	21.42 ✗	21.59 ✗	24.77 ✗
reC15	18.27 ✓	21.74 ✗	24.47 ✗	20.86 ✗	23.12 ✗	24.79 ✗
reC17	19.70 ✓	21.24 ✗	22.72 ✗	21.78 ✗	26.02 ✗	24.80 ✗
reC19	20.77 ✓	23.04 ✗	25.04 ✗	24.73 ✗	21.38 ✗	26.62 ✗
reC23	21.97 ✗	23.45 ✗	26.00 ✗	24.63 ✗	19.63 ✓	27.41 ✗
reC25	23.74 ✗	26.76 ✗	27.84 ✗	27.76 ✗	19.06 ✓	29.90 ✗
reC29	24.01 ✓	25.10 ✗	26.69 ✗	26.08 ✗	25.19 ✗	28.69 ✗
reC33	24.68 ✗	27.61 ✗	29.25 ✗	28.74 ✗	25.84 ✗	31.41 ✗
reC35	24.65 ✓	25.99 ✗	27.25 ✗	26.69 ✗	25.59 ✗	29.76 ✗
reC37	47.82 ✗	56.90 ✗	55.63 ✗	55.10 ✗	46.70 ✓	59.81 ✗
reC39	45.44 ✓	53.40 ✗	54.35 ✗	54.14 ✗	45.58 ✗	56.89 ✗
reC41	47.45 ✗	54.75 ✗	55.85 ✗	65.75 ✗	45.55 ✓	59.18 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

CAAI Trans. Intell. Technol., 2019, Vol. 4, Iss. 3, pp. 148–158

Table 17 Simulation time, in seconds, for the job-shop problems

Instance	Original TLBO	Approach 1	Approach 2	Approach 3	ETLBO	VPTLBO
abz5	444.0 ✓	472.0 ✗	615.7 ✗	546.2 ✗	484.7 ✗	629.2 ✗
abz7	1168.0 ✓	1497.9 ✗	2723.8 ✗	1594.3 ✗	1676.9 ✗	2001.6 ✗
ft06	118.3 ✓	126.6 ✗	164.3 ✗	146.4 ✗	155.3 ✗	165.5 ✗
ft20	410.0 ✓	439.0 ✗	548.7 ✗	471.6 ✗	438.7 ✗	558.0 ✗
la06	293.7 ✓	345.7 ✗	472.3 ✗	394.1 ✗	357.8 ✗	444.7 ✗
la11	429.5 ✗	429.2 ✓	564.6 ✗	499.9 ✗	513.9 ✗	524.1 ✗
la12	387.1 ✗	428.7 ✗	505.3 ✗	519.1 ✗	380.2 ✓	510.9 ✗
la16	379.7 ✓	432.5 ✗	558.9 ✗	573.9 ✗	427.2 ✗	541.8 ✗
la22	648.0 ✓	855.3 ✗	933.8 ✗	759.1 ✗	779.8 ✗	884.5 ✗
la26	901.6 ✓	1059.4 ✗	1326.3 ✗	1148.2 ✗	1178.1 ✗	1327.5 ✗
la31	1247.0 ✓	1467.4 ✗	1647.7 ✗	1670.2 ✗	1748.5 ✗	1768.7 ✗
la35	1392.6 ✓	1452.1 ✗	1901.1 ✗	1640.4 ✗	1398.9 ✗	1768.4 ✗
la36	1033.8 ✗	1129.6 ✗	1209.5 ✗	1167.8 ✗	936.8 ✓	1217.6 ✗
orb05	382.0 ✗	446.8 ✗	566.7 ✗	564.0 ✗	369.0 ✓	561.1 ✗
orb09	389.2 ✓	440.8 ✗	537.3 ✗	482.1 ✗	450.9 ✗	493.9 ✗
swv01	794.8 ✓	1079.3 ✗	1071.5 ✗	1044.0 ✗	906.2 ✗	1136.6 ✗
swv04	830.8 ✓	1055.5 ✗	1123.6 ✗	1020.4 ✗	976.2 ✗	1108.4 ✗
swv05	930.6 ✓	1101.5 ✗	1200.5 ✗	1024.0 ✗	938.5 ✗	1144.4 ✗
swv16	2507.5 ✓	2812.8 ✗	3279.1 ✗	3030.7 ✗	2624.5 ✗	3289.1 ✗
yn1	1675.3 ✗	2323.8 ✗	2288.9 ✗	2553.4 ✗	1601.7 ✓	2193.3 ✗

Symbol ✓ indicates that the result is the best result among all the algorithms. The symbol ✗ indicates that the result was outperformed by at least one algorithm.

Table 18 Friedman test result for simulation time

Statistics	Value
F_C	11.0705
F (flow shop)	77.4857
F (job shop)	70.8571

5.2.3 Simulation time: In this last experiment, our goal is to compare the performance of the proposed approaches with the performance of the original TLBO, ETLBO and VPTLBO in terms of simulation time. For each benchmark instance, we carried out 30 runs of each algorithm. Tables 16 and 17 show the simulation time, in seconds, computed for each algorithm for each benchmark instance of the flow-shop and the job-shop scheduling problems, respectively.

The results presented in Tables 16 and 17 show that the best performance in terms of simulation time in most of the instances for both the flow shop and the job shop was obtained with the original TLBO. This result was expected because the variant versions of TLBO include additional computations that increase the computational cost of the algorithm. The result of the application of the Friedman test considering a significance level $\alpha = 0.05$ is presented in Table 18.

As expected, the null hypothesis H_0 can be rejected for this experiment since the original TLBO presented a better performance in comparison with the variant versions.

6 Conclusions

In this paper, we presented a modified version of the TLBO algorithm. In our proposed version, different weights are assigned to students during the student phase of the algorithm based on the fitness of the current solution of each student, with higher weights being assigned to students with better solutions. The motivation behind this idea is that, in a classroom environment, students are capable of identifying which colleagues have more knowledge about a subject and they tend to invite these colleagues to study together to learn from them.

Three different approaches to assign weights to students were investigated. Numerical simulations using 20 benchmark instances of the flow-shop scheduling problem and 20 benchmark instances of the job-shop scheduling problem were carried out. The proposed approaches were compared with the original TLBO and with two variants available in the literature: the ETLBO and the variable population scheme TLBO. The results show that the proposed approaches outperformed the original TLBO algorithm in terms of convergence speed, with no significant losses in quality of solutions. The original TLBO presented a better performance in terms of simulation time because the additional calculations in the variant versions increase their computational cost. The proposed approaches presented a competitive performance in comparison with other variants of TLBO.

The best performance in terms of convergence speed was obtained with the proposed approach 1, which consists of assigning a fixed weight to half the students with the best solutions and assigning zero to other students. For both the flow-shop and the job-shop scheduling problems, this approach presented the best convergence speed in most of the instances considered in this paper. The application of the Friedman statistical test showed that the increase in convergence speed achieved by the proposed variants is statistically significant. The Friedman test also indicated that the increase in convergence speed did not cause a significant decrease in the makespan. The variant versions of TLBO required more simulation time to run in comparison with the original TLBO. However, the difference in simulation time between the proposed approach 1 and the original TLBO is <1 min for all instances of the flow-shop scheduling problem. For the job-shop scheduling problem, this difference is <2 min in most of the instances. If the algorithm is not used in real-time applications, then the proposed approach 1 is a good alternative to be considered to solve scheduling problems.

Although TLBO has been recently proposed, the number of publications exploring possible modifications to its original form increases rapidly. The TLBO algorithm has been successfully applied to different optimisation problems including combinatorial problems. The results obtained in this paper indicate that searching for the equilibrium between exploration and exploitation in TLBO may lead to additional improvements in the algorithm. As mentioned earlier, all the proposed approaches can be incorporated in most variants of TLBO available in the literature in a straightforward way.

We believe that better results can be achieved by starting the algorithm with the original TLBO and then gradually migrating to one of the proposed approaches. This strategy would strengthen the diversification capability of the method in the first generations while increasing the intensification capability of the method in the last generations. During the first generations, students do not know each other and cannot identify the best students. For this reason, invitations to participate in study groups are more likely to be made randomly. During the execution of the algorithm, the best students become known and invitations to participate in study

groups are driven by this knowledge. However, to implement this strategy, it would be necessary to define a migration rate, which means to add one more input parameter to TLBO. Future research will investigate this topic. Another opportunity to extend this paper is to investigate the performance of proposed approaches in multi-objective optimisation problems.

7 Acknowledgment

The authors acknowledge the support of the Brazilian National Council for Scientific and Technological Development (research fellowship Grant no. 305048/2016-3).

8 References

[1] Rardin, R.L., Uzsoy, R.: 'Experimental evaluation of heuristic optimization algorithms: a tutorial', *J. Heuristics*, 2001, **7**, (3), pp. 261–304
[2] Kennedy, J., Eberhart, R.C.: 'Particle swarm optimization'. Proc. IEEE Int. Conf. Neural Networks, Perth, Australia, 1995, pp. 1942–1948
[3] Holland, J.H.: 'Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence' (MIT Press, Cambridge, MA, USA, 1992)
[4] Dorigo, M.: 'Optimization, learning and natural algorithms'. Politecnico di Milano, Italy, 1992
[5] Rao, R.V., Vakharia, D.P., Savsani, V.J.: 'Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems', *Comput.-Aided Des.*, 2011, **43**, pp. 303–315
[6] Baykasoğlu, A., Hamzadayi, A., Köse, S.Y.: 'Testing the performance of teaching learning based optimization (TLBO) algorithm on combinatorial problems: flow shop and job shop scheduling cases', *Inf. Sci.*, 2014, **276**, pp. 204–218
[7] Rao, R.V., Savsani, V.J., Vakharia, D.P.: 'Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems', *Inf. Sci.*, 2012, **183**, pp. 1–15
[8] Farahani, H.F., Aghaei, J., Rashidi, F.: 'Optimal power flow of HVDC system using teaching–learning-based optimization algorithm', *Neural Comput. Appl.*, 2017, **30**, (12), pp. 3781–3789
[9] Rodrigues, L.R., Gomes, J.P.P., Rocha Neto, A.R., *et al*.: 'A modified symbiotic organisms search algorithm applied to flow shop scheduling problems'. 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 2018, pp. 1–7
[10] Chacko, S., Anand, M.: 'Learning of RBF network by using teaching–learning based optimization (TLBO) algorithm', *Int. J. Eng. Comput. Sci.*, 2017, **6**, (4), pp. 20813–20819
[11] Rao, R.V., Patel, V.: 'An elitist teaching–learning based optimization algorithm for solving complex constrained optimization problems', *Int. J. Ind. Eng. Comput.*, 2012, **3**, pp. 535–560
[12] Chen, D., Lu, R., Zou, F., *et al*.: 'Teaching–learning-based optimization with variable-population scheme and its application for ANN and global optimization', *Neurocomputing*, 2016, **173**, (Part 3), pp. 1096–1111
[13] Zou, F., Wang, L., Hei, X., *et al*.: 'Teaching–learning-based optimization with dynamic group strategy for global optimization', *Inf. Sci.*, 2014, **273**, pp. 112–131
[14] Keesari, H.S., Rao, R.V.: 'Optimization of job shop scheduling problems using teaching–learning-based optimization algorithm', *Opsearch*, 2014, **51**, (5), pp. 545–561
[15] Rao, R.V., Patel, V.: 'An improved teaching–learning-based optimization algorithm for solving unconstrained optimization problems', *Scientia Iranica*, 2013, **20**, pp. 710–720
[16] Chen, D., Zou, F., Li, Z., *et al*.: 'An improved teaching–learning-based optimization algorithm for solving global optimization problem', *Inf. Sci.*, 2015, **297**, pp. 171–190
[17] Chen, D., Zou, F., Wang, J., *et al*.: 'A teaching–learning-based optimization algorithm with producer–scrounger model for global optimization', *Soft Comput.*, 2015, **19**, (3), pp. 745–762
[18] Sabuncuoglu, I., Bayiz, M.: 'Job shop scheduling with beam search', *Eur. J. Oper. Res.*, 1999, **118**, (2), pp. 390–412
[19] Binato, S., Hery, W.J., Loewenstern, D.M., *et al*.: 'A GRASP for job shop scheduling', in Celso, C.R., Pierre, H. (Eds.): 'Essays and surveys on metaheuristics' (Kluwer Academic Publishers, USA, 2000), pp. 59–79
[20] Kiziloz, H.E., Deniz, A., Dokeroglu, T., *et al*.: 'Novel multiobjective TLBO algorithms for the feature subset selection problem', *Neurocomputing*, 2018, **306**, pp. 94–107
[21] Baburao, P., Kumar, R.A., Balamurugan, G., *et al*.: 'A non-dominated sorting TLBO algorithm for multi-objective short-term hydrothermal self-scheduling of GENCOs in a competitive electricity market', *Int. J. Comput. Sci. Eng.*, 2018, **6**, (8), pp. 191–203
[22] Lin, W., Wang, L., Tian, G., *et al*.: 'MTLBO: a multi-objective multi-course teaching–learning-based optimization algorithm', *J. Appl. Sci. Eng.*, 2018, **21**, (3), pp. 331–342
[23] Cheng, T., Chen, M., Fleming, P.J., *et al*.: 'A novel hybrid teaching learning based multi-objective particle swarm optimization', *Neurocomputing*, 2017, **222**, pp. 11–25
[24] Kumar, M., Mittal, M.L., Soni, G., *et al*.: 'A hybrid TLBO-TS algorithm for integrated selection and scheduling of projects', *Comput. Ind. Eng.*, 2018, **119**, pp. 121–130
[25] Dokeroglu, T.: 'Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem', *Comput. Ind. Eng.*, 2015, **85**, pp. 86–101
[26] Rao, R.V.: 'Review of applications of TLBO algorithm and a tutorial for beginners to solve the unconstrained and constrained optimization problems', *Decis. Sci. Lett.*, 2016, **5**, pp. 1–30
[27] Aruna, S., Kalra, S.: 'Review of the teaching learning based optimization algorithm', *Indian J. Comput. Sci. Eng.*, 2017, **8**, (3), pp. 319–323
[28] Mastrolilli, M., Svensson, O.: 'Improved bounds for flow shop scheduling' (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 677–688
[29] Bouzidi, A., Riffi, M.E.: 'Cat swarm optimization to solve flow shop scheduling problem', *J. Theor. Appl. Inf. Technol.*, 2015, **72**, (2), pp. 239–243
[30] Amirghasemi, M., Zamani, R.: 'An effective evolutionary hybrid for solving the permutation flow shop scheduling problem', *Evol. Comput.*, 2017, **25**, (1), pp. 87–111
[31] Šeda, M.: 'Mathematical models of flow shop and job shop scheduling problems', *Int. J. Appl. Math. Comput. Sci.*, 2007, **4**, (4), pp. 241–246
[32] Cruz-Chavez, M.A., Martinez-Rangel, M.G., Hernandez, J.A., *et al*.: 'Scheduling algorithm for the job shop scheduling problem'. Electronics, Robotics and Automotive Mechanics Conf. (CERMA 2007), Morelos, Mexico, 2007, pp. 336–341
[33] Hart, E., Sim, K.: 'A hyper-heuristic ensemble method for static job-shop scheduling', *Evol. Comput.*, 2016, **24**, (4), pp. 609–635
[34] Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C.: 'A hybrid genetic algorithm for the job shop scheduling problem', *Eur. J. Oper. Res.*, 2005, **167**, (1), pp. 77–95
[35] French, S.: 'Sequencing and scheduling: an introduction to the mathematics of the job shop' (John Wiley & Sons, Inc., Chichester, 1987)
[36] Giffler, B., Thompson, G.L.: 'Algorithms for solving production-scheduling problems', *Oper. Res.*, 1960, **8**, (4), pp. 487–503
[37] Rodrigues, L.R., Gomes, J.P.P., Oliveira, S.A.F., *et al*.: 'Improving the computational efficiency of teachingat–learning based optimization for job shop scheduling problems by eliminating unnecessary objective function calls'. Proc. Brazilian Symp. Operational Research (SBPO), Blumenau, Brazil, 2017, pp. 1969–1980
[38] Pandey, H.M.: 'Performance evaluation of selection methods of genetic algorithm and network security concerns', *Procedia Comput. Sci.*, 2016, **78**, pp. 13–18
[39] Montgomery, D.C.: 'Design and analysis of experiments' (John Wiley & Sons, Inc., 2005, 6th edn.)
[40] Friedman, M.: 'The use of ranks to avoid the assumption of normality implicit in the analysis of variance', *J. Am. Stat. Assoc.*, 1937, **32**, (200), pp. 675–701
[41] Friedman, M.: 'A comparison of alternative tests of significance for the problem of *m* rankings', *Ann. Math. Stat.*, 1940, **11**, (1), pp. 86–92
[42] Derrac, J., García, S., Molina, D., *et al*.: 'A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms', *Swarm Evol. Comput.*, 2011, **1**, (1), pp. 3–18