Evolutionary Intelligence (2022) 15:57–79
https://doi.org/10.1007/s12065-020-00487-5

RESEARCH PAPER

A modified teaching learning metaheuristic algorithm with opposite-based learning for permutation flow-shop scheduling problem

Umesh Balande[1] · **Deepti shrimankar**[1]

Received: 2 March 2020 / Revised: 18 July 2020 / Accepted: 8 August 2020 / Published online: 16 September 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

Teaching-Learning-Based Optimization is one of the well-known metaheuristic algorithm in the research industry. Recently, various population-based algorithms have been developed for solving optimization problems. In this paper, a random scale factor approach is proposed to modify the simple TLBO algorithm. Modified Teaching-Learning-Based Optimization with Opposite-Based-Learning algorithm is applied to solve the Permutation Flow-Shop-Scheduling Problem with the purpose of minimizing the makespan. The OBL approach is used to enhance the quality of the initial population and convergence speed. PFSSP is used extensively for solving scheduling problem, which belongs to the category of NP-hard optimization problems. First, MTLBO is developed to effectively determine the PFSSP using the Largest Order Value rule-based random key, so that individual job schedules are converted into discrete schedules. Second, new initial populations are generated in MTLBO using the Nawaz–Enscore–Ham heuristic mechanism. Finally, the local exploitation ability is enhanced in the MTLBO using effective swap, insert and inverse structures. The performance of proposed algorithm is validated using ten benchmark functions and the Wilcoxon rank test. The computational results and comparisons indicate that the proposed algorithm outperformed over five well-known datasets such as Carlier, Reeves, Heller, Taillards and VRF benchmark test functions, compared to other metaheuristic algorithms. The *p*-value indicated the significance and superiority of the proposed algorithm over other metaheuristic algorithms.

Keywords Evolutionary algorithms · Opposite-based learning · Permutation flow-shop scheduling problem · Teaching-learning-based optimization

1 Introduction

Scheduling is a decision-making process, perform vital role in services, manufacturing and production industry. It is traditionally defined as a process of allocating job sequence on different machines that reduces the makespan (completion-time) of a job sequence [1]. Scheduling problems are found in many real-world industries such as textile [2], discrete manufacturing industries [3], electronics [4], chemical [5], production of concrete [6], manufacturing of photographic film [7], iron and steel [8], and internet service architecture [9]. Typically, scheduling problems are classified on the basis of production environments such as single machines, flow-shop, parallel-machines, open-shop, cyclic flow shop and Flexible Job-Shop (FJS). PFSSP is an ultimate engaging research problem in the manufacturing industry with an objective to minimize the total completion time. It is one of the simplified versions of the typical Flow Shop Scheduling Problem (FSSP). The first PFSSP pioneering work was carried out on two machines, with a view to minimize the completion-time [10]. In terms of complexity estimates, PFSSP is found to be NP hard optimization problem [11].

The approaches to solve PFSSP can be broadly grouped into three categories: exact, heuristics and metaheuristic algorithms. The exact-algorithms: dynamic-programming [12], branch and bound techniques [13], and linear-programming [14], that can be successively adopted for solving small problems. However, these techniques do not

✉ Umesh Balande
umeshtbalande@gmail.com

Deepti shrimankar
dshrimankar@cse.vnit.ac.in

1 Visvesvaraya National Institute of Technology, Nagpur 440010, India

produce promising outcomes for large instances in reasonable time. Therefore, heuristic techniques are generally used for large instances. The heuristic techniques are simple, fast and can be used to frame scheduling solutions [15]. A solution can be obtained in stipulated time using the heuristic algorithms, but the produces outcomes may not be optimal. Hence, metaheuristic algorithms are proposed over heuristic algorithms. An existing metaheuristic algorithms for any given problem provide balance between diversification and intensification [16]. There are many nature-inspired algorithms that are used to solve the PFSSP such as genetic algorithm [17], ant colony optimization [18], differential evolution [19], harmony search [20], firefly algorithm [21], bat algorithm [22], cuckoo search [23], TLBO [24], grey wolf algorithm [25], earthworm optimization algorithm [26], monarch butterfly optimization [27], artificial bee colony algorithm [28], jaya algorithm [29], moth search algorithm [30], return-cost-based binary FFA (Rc-BBFA) [31] etc. To overcome the problems faced by single metaheuristic algorithm, recently hybridization of algorithms i.e. combination of two or more local or global search algorithms are used. This also results in increased performance of the metaheuristic algorithms. As the hybridization algorithms can find high-quality feasible outcomes in reasonable time, they are recently used in research development of PFSSP. The advantages for selection of TLBO algorithm, it is easy to implement and easily applicable for practical application. Selection of parameter tune problem is not here. It is not required any mutation and crossover parameters as like genetic algorithm. The performance of TLBO is better than comparative other metaheuristic algorithms. The proper precaution have to take for TLBO algorithm as convergence rate is quickly that is main disadvantage of this algorithm.

In recent decades, researchers are focusing on the hybridization of the genetic algorithm in search of minimum makespan of PFSSP [32]. Hybrid Genetic Algorithm (HGA) [33] was proposed to solve sequence independence of PFSSP. The objective of Particle Swarm Optimization (PSO) for PFSSP, i.e. PSOVNS [34] was to minimize the makespan and the total-flow time. Also, the effective PSO with local search simulated annealing algorithm was used for PFSSP, which balance the exploitation and exploration [35]. Tabu-search-algorithm (TSA) hybridized with improved global search algorithm for solving PFSSP [36]. Hybrid Differential Evolution (HDE) algorithm combined with greedy based local search and Individual Improving Scheme (IIS) was developed to elevate the feasible outcomes quality [37]. The novel Hybrid Cuckoo Search (HCS) was proposed to minimize the makespan and total flow time for solving PFSSP [38]. Enhanced Migrating Bird Optimization (MBO) algorithm was also used to solve a scheduling problem [39]. For minimizing the total flowtime, distributed PFSSP was introduced [40]. While, Improved Migrating Birds Optimization (IMMBO) and a Hybrid-multi-objective Discrete Artificial-Bee-Colony (HDABC) were used to minimize the makespan [41]. Flow shop-scheduling problem to minimize makespan by modified fruit-fly optimization algorithm [42]

Teaching-Learning-Based Optimization (TLBO) was suggested by Rao et al. [24]. It was inspired from the teaching-learning process and it proved to be efficient. The main advantage of using TLBO is that, it is free from any algorithm-parameters. Due to its characteristics, the TLBO algorithm has been gaining recognition in the research industry. The electrochemical discharge machining technique and electrochemical-machining mechanism have been designed using TLBO algorithm [43]. The efficiency of TLBO algorithm was tested on flow-shop and Job-Shop-Scheduling Problem (JSSP) [44]. Shao et al. [45] applied hybrid TLBO algorithm combined with simulated annealing for solving PFSSP. The discrete TLBO algorithm (DTLBO) for solving flowshop rescheduling problem [46]. The probabilistic TLBO algorithm was used to solve no-wait flow-shop scheduling problem (NWFSSO) with high completion-time [47]. The flexible flow shop scheduling problems were solved using the improved TLBO and the JAYA algorithm [48]. The Whale-Optimization-Algorithm (WOA) with a local search approach was also used for solving PFSSP [49].

With this background, the primary contributions of this paper are listed below:

1. MTLBO with OBL method is applied for solving PFSSP. The OBL approach is used to enhance the quality of initial populations (individuals). Using Largest-Order-Value (LOV) rule based random key, MTLBO algorithm is developed to solve PFSSP effectively by changing individuals into discrete job sequences. Nawaz–Encore–Ham (NEH) heuristic method is combined with discrete job sequence to initialize a high quality and a diverse population. Moreover, to enhance the local exploitation ability, the effective swap, insert and inverse structures are included into the MTLBO algorithm.
2. In order to evaluate and analyze the effectiveness of the proposed algorithm, computational experiments are conducted using ten benchmark test functions and Wilcoxon signed-rank test.
3. Comprehensive extensive experiments are conducted using different well-known datasets such as Carlier, Reeves, Heller, Taillards and VRF benchmark test functions.
4. To validate its performance, the MTLBO algorithm is compared with various well-known metaheuristic algorithms.

The structure of the paper is as follows: Sect. 2 presents the description of the PFSSP. Section 3 provides the related work of basic TLBO algorithm. Section 4 describes the proposed algorithm MTLBO. Section 5 presents computational results and discussion and at last Sect. 6 discusses the conclusions. The symbols used to describe the PFSSP and along with their explanations are shown in Table 1.

2 The description of the PFSSP

In the PFSSP, the set of n jobs ($j = j_1, j_2, \dots, j_n$) are consecutively handled on the set of m machines ($M = M_1, M_2, \dots, M_m$). As each machine can process only one job at a time, the handling time for each job is determined as $P_{i,j} = (i = 1, 2, \dots., n, j = 1, 2, \dots, m)$. Each job is given a task or operation on each machine and is represented as $T_j = T_{j1}, T_{j2}, \dots, T_{jm}$. The sequence followed by each job over each machine is the same. A schedule Π is denoted as a permutation ($\Pi = \pi_1, \pi_2, \dots, \pi_n$), where Π represents set of all permutations for n jobs. Let us assume $Z(\pi_i, m)$ to be the makespan of job (π_i) over the machine m. S_{1,π_i-1,π_i} indicates initial time of job-permutation. Let us assume that $P_{\pi_i,j}$ represents the handling time of job (π_i) over the machine j. Hence, the PFSSP can be mathematically represented as follows:

$$Z(\pi_1, 1) = p_{\pi_1,1} \tag{1}$$

$$Z(\pi_i, 1) = Z(\pi_{i-1}, 1) + p_{\pi_i,1} + S_{1,\pi_i-1,\pi_i}, \quad i = 2, 3, \dots., n \tag{2}$$

$$Z(\pi_i, j) = Z(\pi_1, j-1) + p_{\pi_1,j}, \quad j = 2, 3, \dots., m \tag{3}$$

$$Z(\pi_i, j) = max(Z(\pi_{i-1}, j) + S_{j,\pi_i-1,\pi_i}, (Z(\pi_i, j-1)) + p_{\pi_i,1} \quad i = 2, 3, \dots., n; \; j = 2, 3, \dots., m \tag{4}$$

Table 1 Symbols used

Symbols	Explanation
n	Indicate the total amount of jobs
m	Indicate the as total number of machines
i	Machine
j	Job
$P_{i,j}$	Total time required for job completion
T_i	Task on each machine
Π	Set of all permutations of n jobs
$Z(\pi_i, m)$	Makespan of job π_i on machine m
P	PFSSP
Z_{max}	Objective to minimize the completion time
$Z(\pi^{opt})$	Optimal permutation

where n indicates the total number of jobs, m denotes the machines, P represents PFSSP and Z_{max} is the goal to minimize makespan of the last job on the last machine.

In PFSSP, main goal is minimization of the maximum makespan, which can be determined as

$$Z_{\max}(\pi) = Z(\pi_n, m) \tag{5}$$

The main goal of PFSSP is to determine the optimal permutation from the set of all permutations:

$$Z(\pi^{opt}) \le Z(\pi_n, m) \quad \text{for all } \Pi \tag{6}$$

where π^{opt} denotes the optimal permutation.

3 Basic TLBO algorithm

Teaching-learning process is widely known since the ancient times, where one individual (learner) tries to learn from other individual. It has been applied in optimization of two thermoelectric coolers [50], cluster data [51] and image processing [52].

TLBO [24]is a population-based metaheuristic algorithm, that has two main fundamental phases: a teacher-phase and the learner-phase. In teacher phase of the algorithm, teacher teaches or shares their knowledge to learners (students). In the learning phase, the group of students or learners enhance their knowledge through teacher or interaction among themselves. Figure 1 illustrates the schematic flow of basic TLBO algorithm.

3.1 Teacher-phase (TP)

It invokes the exploration phase of TLBO where students learn from the teacher. In this stage, teacher attempts to enhance the mean result of the class. For the objective function $f(Z)$ with N-dimensional variable, $Z = (z_{i,1}, z_{i,2}, \dots, z_{i,N})$ represents the position of ith learner (or student) solution for N-dimensional problem. Thus, the class mean position with P learners (where P is the size of population) is represented as

$$Z_{mean} = \frac{\sum_{i=1}^{P} z_{i,1}, \sum_{i=1}^{P} z_{i,2}, \dots, \sum_{i=1}^{P} z_{i,N}}{P} \tag{7}$$

The students with the optimal value (i.e. best value) in current generation are represented as $Z_{teacher}$. The position of each student is shown in Eq. (8)

$$Z_{i,new} = Z_{i,old} + rand(Z_{teacher} - T_F.Z_{Mean}) \tag{8}$$

where $Z_{i,new}$ and $Z_{i,old}$ are the ith students (learners) new and old positions respectively. $Rand$ is dispersed random number in the range of [0, 1]. T_F is teacher-factor which decided

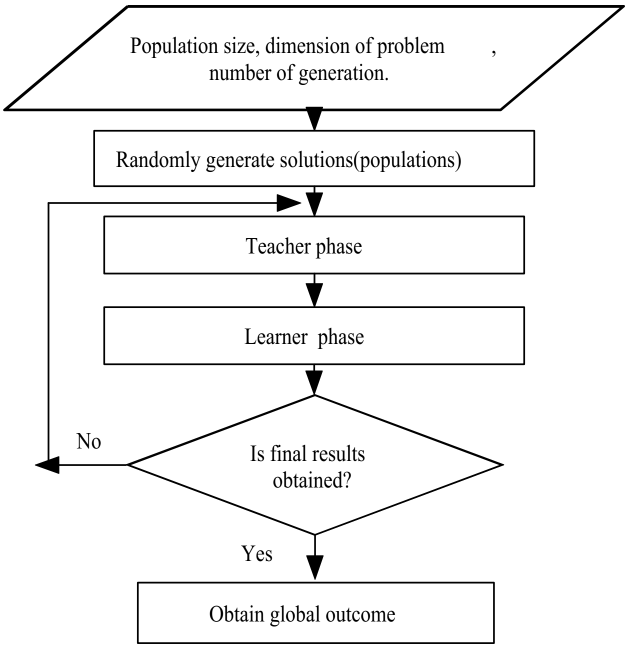


Fig. 1 The schematic flow diagram of basic TLBO

average value to be altered. T_F can be any 1 or 2, it can be generated randomly using Eq. (9) as

$$T_F = round[1 + rand(0, 1)\{2 - 1\}] \tag{9}$$

where T_F is achieved randomly when the TLBO in range of [1–2], where "1" represent to no raise in proficiency-level and "2" represents to entire transmission of knowledge.

3.2 Learner-phase (LP)

The learner-phase is an exploitation phase of TLBO algorithm, where a learner randomly collaborates with other learners to improve his/her proficiency. Student (Learner) X_i, learns new things if another student $X_j (j \neq i)$ has more proficiency than him/her. Learning process can be depicted mathematically as

$$Z_{i,new} = \begin{cases} Z_{i,old} + rand * \left(Z_i - Z_j\right), \\ \text{if} \quad f(Z_i) \leq f(Z_j). \\ Z_{i,new} = Z_{i,old} + rand * \left(Z_i - Z_j\right), \\ \text{if} \quad f(Z_i) > f(Z_j). \end{cases} \tag{10}$$

where *rand* is a random number in the range of [0, 1], $Z_{i,new}$ it gives the preferable feasible value.

3.3 Duplication elimination

A duplicate solution should not be kept in the population as it may cause premature convergence of the algorithm [53, 54]. The duplication elimination strategy is given in Fig. 2.

4 Proposed MTLBO approach

In this section, the proposed MTLBO is explained with the local search strategy for makespan minimising the makespan in PFSSP. The proposed algorithm flowchart is given in Fig. 3. The symbol and meaning of the proposed algorithm is shown in Table 2.

4.1 Representation of solution for MTLBO algorithm

The basic TLBO algorithm was initially proposed for solving different continuous-optimization examples. However, the PFSSP is a discrete optimization technique. The basic TLBO algorithm, therefore cannot be used to solve PFSSP directly. So, the basic TLBO has to be modified using the

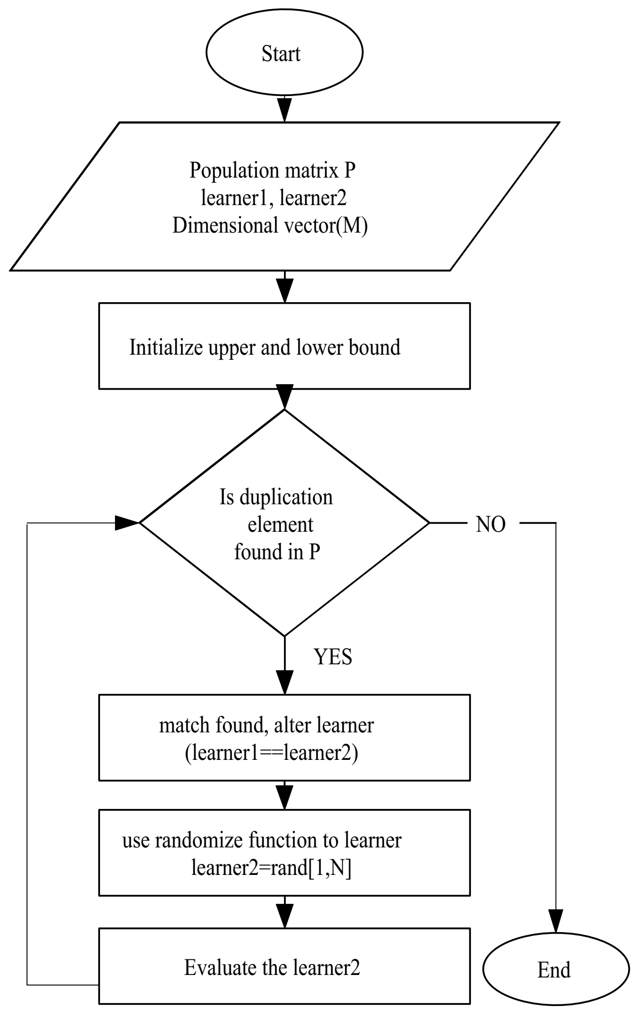


Fig. 2 Elimination of duplicate solution in TLBO algorithm

concept of differential evolution with random scale factor (*DERSF*) [55]. This term is scaled by scale factor (R) in a random way in the range (0.5, 1) and it is given as

$$R = 0.5(1 + rand(0, 1)) \tag{11}$$

To apply MTLBO to PFSSP, one of the key concerns is to create mapping rule between job sequence and the vector of individuals. To convert individual $Z_i = [z_{i,1}, z_{i,2}, \dots, z_{i,n}]$ into job permutation $\Pi_i = [\pi_{i,1}, \pi_{i,2}, \dots, \pi_{i,n}]$, the following random keys (called as robust representation [56]) are used namely the Smallest Positive Value (*SPV*) [34], Largest Ranked Value (*LRV*) [57] and Largest Order Value (*LOV*) [58]. The *SPV*, *LRV* and *LOV* rules are used in existing research [38]. The job sequence relationship scheme plays a vital role in the developmental mechanism of the proposed MTLBO. If exchange of job sequence is not proper in PFSSP, then it will increase the computational time of the proposed algorithm.

In this study, *LOV* rule is utilized. *LOV* is a very simple method, which is focused on random-key presentation. In *LOV* rule, by ranking the individual of $Z_i = [z_{i,1}, z_{i,2}, \dots, z_{i,n}]$ with decreasing order, temp job permutation sequence is generated as $\sigma_i = [\sigma_{i,1}, \sigma_{i,2}, \dots, \sigma_{i,n}]$. Then a job sequence permutation is obtained as $\Pi_i = [\pi_{i,1}, \pi_{i,2}, \dots, \pi_{i,n}]$ using Eq. (12).

$$\pi_{i\sigma_{i,j}} = j \tag{12}$$

where $j \in [1, \dots, n]$. In Table 3, *LOV*-rule is described with an easy example ($n = 6$) where the solution representation of individual vector (Z_i) is obtained with job dimension, position (location) and job permutation.

For Z_i= [0.6, 0.12, 1.8, − 0.48, 1.7, − 1.06], it can be seen that $z_{i,3}$ has the largest value, and so it is selected as the first order of a job permutation. After that $z_{i,5}$, $z_{i,1}$, $z_{i,4}$ and $z_{i,6}$ are preferred as job-permutation. Hence, the job sequence Π_i = [3, 5, 1, 2, 4, 6] is obtained. From this formulation, the conversion using *LOV* rule is clear, which makes MTLBO suitable for solving PFSSP.

Table 2 Symbols and meaning used in proposed algorithm

Symbols	Meaning
NP	Population size of proposed algorithm
MaxG	Maximum generation or iteration
T	Maximum CPU time
Z_{new} and Z_{new}	Old and new iteration in population
OBL	Opposite Based Learning.
GOBL	Generalized Opposite-Based Learning
NEH	Nawaz–Enscore–Ham
Z_i	Design variable of problem
π_i	Job sequence permutation
LOV	Largest Order Value
DERSF	Differential Evolution with Random Scale Factor

4.2 Population initialization

This phase plays a vital role and it is applied uniformly and randomly. A population vector $Z_i = [z_{i,1}, z_{i,2}, \dots, z_{i,n}]$ is randomly generated. Nawaz–Enscore–Ham (*NEH*) method is used for producing good initial population. *NEH* is the heuristic method to obtain optimal solution to the PFSSP. The *NEH* steps are described below:

Step 1: All jobs are arranged in non-increasing procedure based on their overall handling time on all machines. Then job permutation $\Pi(i) = [\pi(1), \pi(2), \dots, \pi(3)]$ is achieved.

Step 2: Select any two jobs permutations, for examples $\pi(1)$ and $\pi(2)$. Then, all forms of permutation records of these two jobs are calculated and then an optimal order is taken.

Step 3: Select any job permutation $\pi(j)$,(j=3,4,....,n) and determine best permutation of jobs by assigning the available position to which they had scheduled. The optimal arrangement is selected from the next iteration.

The outcome created by *NEH* algorithm is a discrete job-permutation. To implement the MTLBO algorithm, the job permutation sequences are converted into individual by using Eq. (13), the conversion process is performed as:

$$Z_{NEH,i} = Z_{min,i} + \frac{\left(Z_{max,i} - Z_{min,i}\right)}{n} \times \left(I_{NEH,i} - 1\right) \qquad i = 1, 2, \dots, n \tag{13}$$

where $I_{NEH,i}$ is job-index in ith dimension of job-permutation (Π). $Z_{max,i}$ and $Z_{min,i}$ are maximum and minimum bounds. $Z_{NEH,i}$ is the individual value of given problem at the ith dimension.

Let us consider that all job-permutation (Π) achieved by NEH heuristic approach are given as Π = [2, 6, 3, 5, 1, 4]. Hence, the index of first carry on job sequence is given as value 5, index of second carry on job sequence is given as value 1. The index sequence ($I_{NEH,i}$) and individual value ($Z_{NEH,i}$) are shown in Table 4.

4.3 Opposition based learning

Many optimization techniques are developed using OBL. OBL is used to enhance the performance and search ability to find a solution of population based algorithms [59]. Rahnamayan et al. [60] first proposed OBL strategy in an optimization method. The main purpose of OBL is to choose

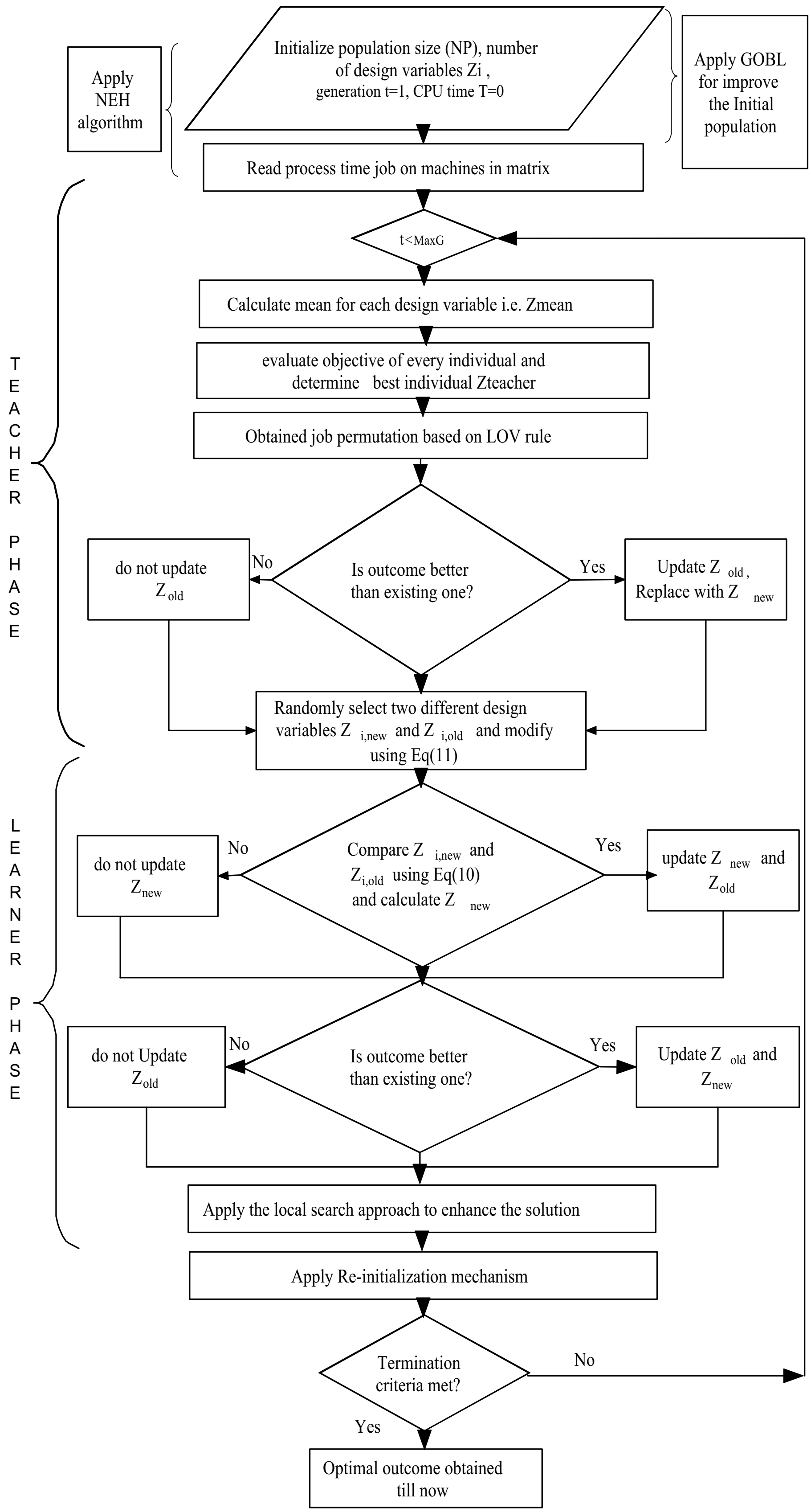


Fig. 3 Proposed MTLBO algorithm

Table 3 Individual solutions represented in LOV-rule

Dimension (D)	1	2	3	4	5	6
Location	0.6	0.12	1.8	− 0.48	1.7	− 1.06
Job	3	5	1	2	4	6

a better current candidate outcome by simultaneous evaluation of the current outcomes and its opposite outcomes. Mainly, OBL is applied to any population based algorithm during two phases: initial population and evolutionary phase. We can improve the initial population of proposed MTLBO algorithm using OBL. The concept of OBL depends on opposite number and opposite point.

Opposite number : Let a $z \in [x, y]$ be any real number $\mathbb{R}$. its opposite number z^{opp} is given by

$$z^{opp} = x + y - z \tag{14}$$

Opposite point: Let $Y(z_1, z_2, \dots, z_n)$ be point in N- dimensional space where $z_i = [x, y]$;. The opposite point of opposite population $(z_1^{opp}, z_2^{opp}, \dots, z_n^{opp})$ is given as:

$$z_1^{opp} = x_i + y_i - z_i \tag{15}$$

4.3.1 GOBL optimization in MTLBO algorithm

The convergence rate of MTLBO algorithm can be improved with the help of GOBL. Wang et al. [61] proposed Generalized OBL model (GOBL). The proposed MTLBO approach uses this method to improve the performance of algorithm. When the premature convergence and stagnation state are identified, re-initialization method (i.e. OBL concept) is stimulated to avoid premature convergence in MTLBO algorithm.

4.4 Local search approach

The local search approach is applied to improve the quality of best scheduling (let B^*) such as

(a) In the job permutation, each job in the B^* may have variable dimension d, where move the job sequence to all other $(d-1)$ location.
(b) For each movement, calculate the new schedule of the job permutation, i.e. consider B^*= new schedule.
(c) Least search probability (LSP) is used for local search approach.

By using the Kaveh et al. [62], the probability value P is changed to each generation(iteration) such as

$$P = 0.3\left(\frac{iteration}{\max\_iteration}\right) \tag{16}$$

where *iteration* is current iteration and *max_iteration* is total number of iterations required to perform the experiments. Thus, changing the value of P improves the solution of the outcomes.

4.5 Swap, insert and inverse operators

To enhance the performance and quality of solution for MTLBO, some operators are chosen such as swap, insert and inverse. These operators would enhance local search ability [63] and improve the outcome solution. Let us randomly take any two different positions m and n in a job permutation (π), if $m < u$, m will be inserted at the back of u. Otherwise, m is to be inserted in front of u. The details of these structures are as follows:

4.5.1 Swap operator

The Fig. 4 shows the swap operation. Randomly select any two separate locations from a job-permutation (π) such as m and u and swap them.

4.5.2 Insert operation

In Fig. 5, randomly select any two separate locations from a job permutation. Here job permutation (π) is removed from position m and inserted into other position u.

Table 4 Transformation of the job-permutation (Π) to the individual vector

Dimension (D)	1	2	3	4	5	6
Permuatation by NEH	2	6	3	5	1	4
job sequence ($I_{NEH,i}$)	5	1	3	6	4	2
individual value ($Z_{NEH,i}$)	0.6	− 1.8	− 0.6	1.2	0	− 1.2

4.5.3 Inverse operation

In Fig. 6, randomly change the subsequence between two separate locations of a job permutation (π). By performing the inverse operation, jobs in solution π and π_{new} are interchanged.

4.6 Re-initialization structure

The experiment is conducted on the proposed algorithm, local search is used for fast convergence of the outcomes. Thus, at some extent the algorithm outcomes are trapped in local optimum i.e. premature convergence may occur. To overcome this situation, re-initialization mechanism is developed. If the feasible optimal solution is not found after 40 successive generation, then reinitialize the learners (populations). The re-initialization scenario is developed and it is given as: half of the populations are exchanged with the best populations achieved in the earlier process and another half of the populations are generated randomly.

4.7 Computational-complexity of MTLBO in term of big O notation

Let there be n jobs and m machines in PFSSP, the size of population is p. Generally in EA the computation complexity determine as of $O(d * p + fit * p)$, d is dimension of problem, fit is fitness or cost function (here minimization of maximum makespan is cost function). In the Initial stage of MTLBO computational complexity of NEH O() and remaining population generated by randomly using OBL technique is $O(0.5 * n^2 mp)$. The computational complexity at initial phase is $O(n^2mp) + O(0.5 * n^2mp)$. In teaching phase, due swap and local search computational

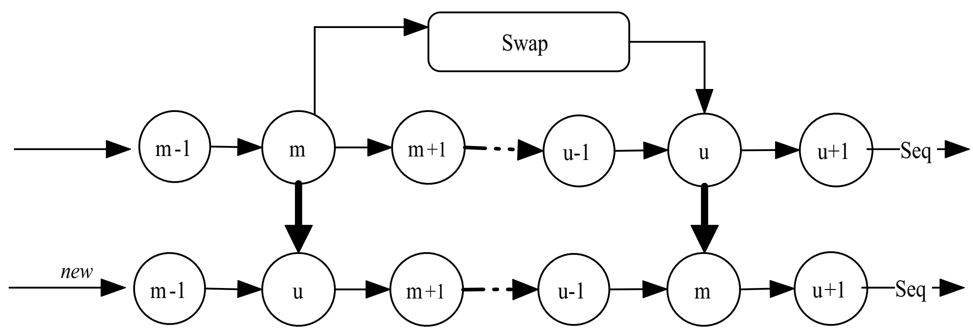


Fig. 4 Local-search using swap operator

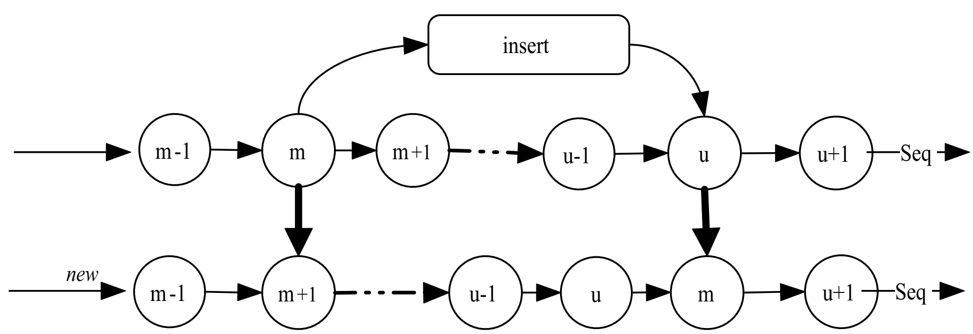


Fig. 5 Local-search using insert operator

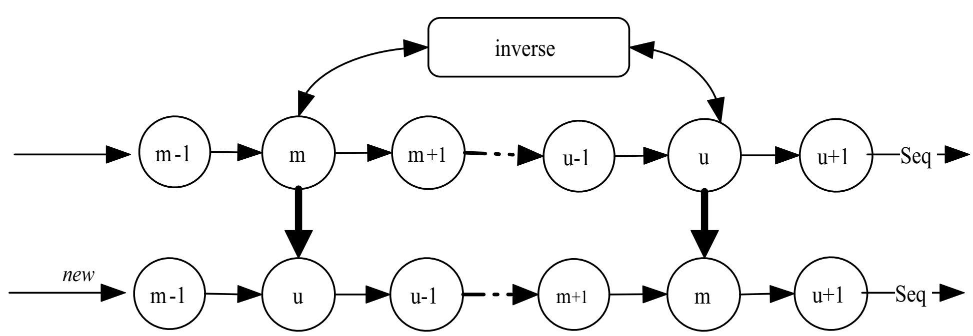


Fig. 6 Local-search using inverse operator

complexity is $O(n^2mp)$. In learning phase, computational complexity is $O(n^2m/2)$, due the learner gets information from teacher or exchange information themselves.

Total computation complexity of MTLBO for computing PFSSP :

$O(n, m, p) = O(n^2mp) + O(0.5 * n^2mp) + O(n^2mp) + O(0.5 * n^2mp) + O(n^2m/2) \approx O(n^2mp) = O(50 \times 1000\, n^2) = O(50000\, n^2)$

5 Computational results and comparisons

This section represents the computational outcomes generated from the MTLBO algorithm. The MTLBO is tested over ten standard benchmark test functions which are taken from Liang [64]. These test functions are either unimodal or multimodal. The tuning for various performance measures are calculated from various evolutionary algorithms by the best, worst, mean (average) and SD (standard-deviation) outcomes [65]. The unimodal benchmark functions validate accuracy and the convergence speed of proposed algorithm whereas multimodal benchmark test functions validate using the global search ability of proposed algorithm. Then, the computational results of MTLBO algorithm are analyzed using Wilcoxon signed rank test method. There are various kinds of stopping criteria in the literature of NIAs, like maximum-number of iterations (G) , number of function evaluation ($NFEs$), limit of time and tolerance. The experiment uses G (i.e. number of generations) as stopping criteria for proposed MTLBO algorithm.

5.1 Experimental setup

In the following computational experiments, the performance and efficiency of MTLBO algorithm is applied on extensively used benchmark functions. There are mainly five well known benchmark functions such as

(a) Carlier datasets (contains eight instances) consist of eight combinations, ranging from 11 jobs with 5 machines to 8 jobs with 8 machines [66].
(b) Reeves and Yamada benchmark datasets (contains twenty-one instances) consist of seven combinations, ranging from 20-jobs with 5-machines to 75-jobs with 20-machines [67].
(c) Heller datasets (contains two instances) consist of two combinations, ranging from 20 jobs with 10 machines to 10 jobs with 10 machines [68].
(d) Taillard datasets (contains 120 instances) consist of twelve combinations, ranging from 20 jobs with 5 machines to 500 jobs with 20 machines [69].
(e) VRF benchmark functions [70] (each contain 240 small and 240 large instances, which have upto 800 jobs on 60 machines) data is available at http://soa.iti.es.

The performance measure indicators for benchmark functions are $ARPD$ (Average Relative Percentage Deviation), $BRPD$ (Best Relative Percentage Deviation) and $WRPD$ (Worst Relative Percentage Deviation) [71]. They are calculated in Eqs. (17), (18), and (19) respectively.

$$ARPD = \frac{Z_{best} - Z_{opt}}{Z_{opt}}.100 \tag{17}$$

$$BRPD = \sum_{i=1}^{n} \left(\frac{Z_i - Z_{opt}}{Z_{opt}} \right) \frac{1}{n} \times 100 \tag{18}$$

$$WRPD = \frac{\left(Z_{worst} - Z_{opt}\right) \times 100}{Z_{opt}} \tag{19}$$

where Z_{opt} is best makespan value, Z_{best}, Z_i and Z_{worst} are makespan solutions obtained in tested algorithms. Smaller value of $ARPD$ indicates that the algorithm are better.

In Table 5, the given parameters are calibrate for MTLBO by conducting several experiments. All the experimentation and analysis of MTLBO algorithm have been implemented on an Intel 3.40 GHz core $i5$ processor where MTLBO algorithm was programmed with Matlab 8.4 (R2014b) under Win7(x64) with 8 GB RAM.

5.2 Testing proposed algorithm using benchmark functions

In this experiment, ten unconstrained benchmark functions are tested using MTLBO algorithm for population size ($NP = 50$) and dimension ($D = 50$). The performance of the MTLBO is calculated by 25 independent runs over the given ten benchmark functions. In Table 6, the effective results of MTLBO are shows in terms of performance measures such as best, worst, mean and SD values on different population based algorithms.

5.2.1 Diversification and intensification of proposed algorithm

The unimodal optimization function (for e.g. Fun2 : Schwefels) is used to find the balance between diversification

Table 5 Parameter used for experiments of MTLBO algorithm

Parameter	Value	Remarks
Population size (*NP*)	50	The population size 50 is selected after several experiments
Stopping criteria (i.e. here number of generations)	1000	Maximum number of iterations used as stopping criteria for algorithm
Number of runs of each problem	25	It is used for finding best possible value for a problem
Z_{opt}		Best mean makespan found so far
LSP	0.01	Least search probability value for all benchmark function such as Carlier, Reeves, Heller and Taillard

(exploration) and intensification (exploitation). In the Fig. 7 indicate the convergence of diversity of population with iterations for unimodal optimization function (i.e schwefels function). This is applied for five metaheuristic algorithms (i.e. GA, PSO, DE, TLBO and MTLBO). In the Fig. 7, the graph indicates that proposed algorithm MTLBO decrease gradually during the exploration, hence global optimal value obtained. The solution obtained during exploitation of algorithm is precise and stable.

5.2.2 Statistical analysis of proposed algorithm

To investigate the MTLBO algorithm, statistical procedure is applied to perform analysis such as parametric and non-parametric tests. The parametric test is based on assumptions made during the analysis of an experiment in computational intelligence. On the other hand, the non-parameter test is applied when the previous conclusions are not satisfied. Usually, this test deals with nominal or ordinal data, but it can also be applied to continuous data by performing the ranking based transformation.

The non-parametric test is applied to investigate the proposed algorithm. There are two types of non-parametric tests such as pair-wise comparison and multiple comparisons. To perform analysis of the proposed algorithm, pair-wise comparison is applied to MTLBO. In pair-wise procedure, comparison is made between two methods (algorithms), by obtaining *p*-value from one another. Wilcoxon signed rank test is a powerful non-parametric test for the pair-wise procedure as given in [72].

In Table 7, results of Wilcoxon signed rank test for MTLBO algorithm, with different metaheuristic algorithms over ten benchmark functions are shown. The test is conducted using 25 independent runs of the proposed algorithm, which differs significantly from other algorithms. Table 7 records the *p*-value for ten unconstrained benchmark functions with $R+$ and $R-$ values. The $R+$ value indicate a sum of rank in which the first algorithm outperform compare to the second algorithm. $R-$ value indicate a sum of ranks second algorithm outperforms compared to the first algorithm. The non significant *p*-value is represented in bold in Table 7.

From the Table 7, it is clear that the performance of the MTLBO algorithm is significantly better than other metaheuristic algorithms. Functions $Fun1$, $Fun2$, $Fun3$, $Fun5$, $Fun7$ and $Fun10$ have significant *p*-value ($p \leq 0.05$), in MTLBO versus DE, whereas functions $Fun4$, $Fun6$ and $Fun9$ does not have significant value.

5.3 Comparison of MTLBO algorithm on Carlier benchmark datasets

The effectiveness of MTLBO is compared with various metaheuristic algorithms on Carlier benchmark datasets. Table 8, shows Carlier datasets range from Car01 to Car08 with size range from 11×5 to 8×8. The performance of MTLBO is calculated using *BRPD*, *ARPD* and *WRPD* along with best makespan value (Z_{opt}). The MTLBO algorithm is compared with L-HDE [37], PSOVNS [34], DBA [73], HBSA [74], and PSOMA [35].

From the Table 8, it is observed that performance of proposed algorithm is better than given population based algorithm. The effective values of MTLBO algorithm shows better performance by using parameters of *BRPD*, *ARPD*, *WRPD* and best makespan values.

5.4 MTLBO algorithm on Reeves and Heller benchmark datasets

The effectiveness of proposed algorithm is compared with Reeves and Heller datasets. In the Table 9 indicates Reeves datasets range from REC001 to REC041 with size range from 20× 5 to 75× 20. The performance of MTLBO is calculated using *BRPD*, *ARPD* and *WRPD* along with best makespan value (Z_{opt}). The MTLBO algorithm is compared with L-HDE, PSOVNS, DBA, HBSA, PSOMA, and HCS [34]. Note that, the best values are shown in bold.

From Table 9, it is clear that the efficiency of MTLBO is superior than given population based algorithm. The effective values of MTLBO algorithm shows better performance by using parameters of *BRPD*, *ARPD*, *WRPD* and best makespan values.

Table 6 The comparison results of GA, PSO, DE, HS, TLBO, MTLBO (D = 50)

Functions	Parameters	GA	PSO	DE	HS	TLBO	MTLBO
*Fun*1	Best	0	1.000E−08	1.000E−08	1.000E−08	0	0
	Worst	0	1.000E−08	1.000E−08	1.000E−08	0	0
	Mean	0	1.000E−08	1.000E−08	1.000E−08	0	0
	SD	0	1.000E−08	1.000E−08	1.000E−08	0	0
*Fun*2	Best	1.744E+05	1.145E+05	2.070E+05	7.753E+05	1.088E+05	**5.863E+04**
	Worst	1.050E+06	8.660E+05	9.740E+05	9.247E+05	2.063E+05	**9.088E+04**
	Mean	4.760E+05	3.063E+05	4.720E+05	8.737E+05	1.684E+05	**7.437E+04**
	SD	2.138E+05	4.192E+05	1.630E+05	5.234E+05	1.850E+04	**1.023E+04**
*Fun*3	Best	2.553E+06	2.104E+05	9.010E+03	1.608E+04	5.415E+03	**3.102E+03**
	Worst	8.438E+08	3.655E+06	1.990E+04	3.718E+04	1.235E+04	**6.113E+03**
	Mean	1.057E+08	8.226E+05	7.880E+03	2.781E+04	7.084E+04	**3.543E+03**
	SD	1.493E+08	7.674E+05	2.830E+03	6.128E+03	3.594E+04	**1.954E+03**
*Fun*4	Best	4.901E−01	3.450E+02	3.230E+02	1.460E+04	2.971E+02	**1.520E+02**
	Worst	3.450E+01	2.250E+02	3.660E+03	5.031E+04	8.562E+02	**4.163E+02**
	Mean	3.330E+00	9.923E+01	1.380E+03	2.983E+04	6.578E+02	**3.766E+02**
	SD	4.883E+00	5.447E+01	8.140E+02	1.007E+04	1.760E+02	**8.700E+01**
*Fun*5	Best	1.000E−08	1.000E−08	0	1.000E−08	1.000E−08	**0**
	Worst	1.000E−08	1.000E−08	0	1.000E−08	1.000E−08	**0**
	Mean	1.000E−08	1.000E−08	0	1.000E−08	1.000E−08	**0**
	SD	1.000E−08	1.000E−08	0	1.000E−08	1.000E−08	**0**
*Fun*6	Best	3.657E+01	2.060E+01	4.340E+01	3.293E+01	1.750E+01	**8.050E+00**
	Worst	9.707E+01	8.856E+01	4.340E+01	5.563E+01	4.337E+01	**2.763E+00**
	Mean	4.723E+01	5.427E+01	4.340E+01	4.457E+01	4.073E+01	**2.963E+00**
	SD	1.402E+01	2.034E+01	4.340E+01	5.984E+00	6.959E+00	**2.390E−01**
*Fun*7	Best	1.513E+01	5.600E+01	6.730E−02	5.995E+02	3.074E+01	**1.065E+01**
	Worst	1.043E+02	1.770E+02	9.090E+00	1.301E+03	5.547E+01	**2.198E+01**
	Mean	4.165E+01	1.670E+02	9.960E−01	9.090E+02	4.963E+01	**2.397E+01**
	SD	1.834E+01	5.518E+02	1.940E+00	2.196E+02	5.314E+00	**2.764E+00**
*Fun*8	Best	2.307E+01	2.111E+01	2.100E+01	4.553E+01	2.099E+01	**9.674E+00**
	Worst	2.425E+01	5.012E+00	2.120E+01	6.189E+01	2.110E+01	**1.199E+01**
	Mean	2.119E+01	3.606E+00	2.110E+01	5.396E+01	2.107E+01	**9.760E+00**
	SD	3.985E−02	7.355E−01	4.150E−01	1.680E−01	2.715E−02	**1.375E−02**
*Fun*9	Best	5.212E+01	9.598E+01	1.870E+01	9.339E+01	5.644E+01	**2.287E+01**
	Worst	7.770E+01	1.333E+02	7.110E+01	8.447E+02	6.255E+01	**3.186E+01**
	Mean	7.431E+01	1.136E+02	2.730E+01	6.154E+01	6.089E+01	**3.053E+01**
	SD	3.967E+00	8.869E+00	7.550E+00	1.260E+01	1.661E+00	**6.530E−01**
*Fun*10	Best	2.713E−02	9.560E−03	1.930E−07	8.966E−03	7.396E−03	**3.452E−03**
	Worst	4.385E−01	1.684E−02	4.440E−02	7.937E−02	2.710E−02	**1.721E−03**
	Mean	1.047E−01	9.840E−02	1.700E−02	5.664E−02	1.763E−02	**8.760E−03**
	SD	7.093E−02	9.845E−02	1.140E−02	9.903E−03	6.850E−03	**2.360E−03**

The comparison of the mean value of *BRPD*, *WRPD* and *ARPD* for the dataset Reeves is shown in (Figs. 8, 9, 10). MTLBO is compared with various population-based algorithms, it can be seen that MTLBO algorithm achieves best outcomes. In mean value of *BRPD*, the largest value PSOVNS is 2.1665. In average value of *ARPD*, the largest value on DBA is 2.5875. In mean value *WRPD*, the largest value on DBA is 3.346. MTLBO shows best and effective outcomes as compared to other population based algorithms. Hence, MTLBO is most reliable and effective algorithm for solving PFSSP.

In Table 10, Heller datasets have two instances such as Hel1 and Hel2 with the size range 20×10 and 10×10. Hamdi et al. [75] determined the upper and lower bound of

Table 7 Wilcoxon signed rank test for *p*-value comparing MTLBO with various EAs over ten benchmark functions ($p \leq 0.05$)

Funtions	Parameter	MTLBO versus GA	MTLBO versus PSO	MTLBO versus DE	MTLBO versus HS	MTLBO versus TLBO
*Fun*1	*p*-value	8.31E−05	8.06E−05	6.20E−04	8.34E−05	9.80E−04
	R+	210	210	188	210	210
	R−	68	68	79	68	79
*Fun*2	*p*-value	8.31E−05	8.31E−05	1.20E−04	8.31E−05	3.20E−04
	R+	210	210	156	210	168
	R−	68	68	74	68	71
*Fun*3	*p*− value	3.04E−02	1.43E−02	2.20E−04	2.22E−03	2.41E−03
	R+	156	168	156	122	122
	R−	74	71	74	83	83
*Fun*4	*p*−value	3.99E−02	3.99E−02	6.56E−01	3.99E−02	3.99E−02
	R+	168	168	119	168	168
	R−	71	71	206	71	71
*Fun*5	*p*-value	8.31E−05	8.31E−05	5.50E−04	8.34E−05	9.80E−04
	R+	210	210	156	210	156
	R−	68	68	74	68	74
*Fun*6	*p*-value	9.94E−03	9.94E−03	**6.09E−01**	9.94E−03	NA
	R+	122	122	119	122	NA
	R−	83	83	206	83	NA
*Fun*7	*p*-value	8.31E−05	8.31E−05	8.31E−05	8.31E−05	8.31E−05
	R+	210	210	210	210	210
	R−	68	68	68	68	68
*Fun*8	*p*-value	8.31E−05	1.52E−03	3.13E−03	8.31E−05	8.31E−05
	R+	210	122	126	210	210
	R−	68	83	145	68	68
*Fun*9	*p*-value	8.31E−05	8.31E−05	**1.44E−01**	9.60E−04	4.30E−03
	R+	210	210	126	156	122
	R-	68	68	145	74	83
*Fun*10	*p*-value	8.31E−05	8.31E−05	7.35E−02	8.31E−05	4.60E−03
	R+	210	210	126	210	122
	R−	68	68	145	68	83

these datasets. The MTLBO is also compared with NEH, DSOMA [76] and SG [11]. It can be seen from the outcomes that MTLBO achieves better value for upper bound for Hel1 and Hel2 problems.

5.5 MTLBO algorithm on Taillard benchmark dataset

The comparison of the MTLBO with basic TLBO and other metaheuristic algorithms are carried out using Taillards benchmark functions [69]. The problem size of Taillard benchmark functions range from 20×5 to 500×20. The minimum and maximum bounds of Taillard's test function are given in OR-Library (i.e. Eric Taillard's webpage).

5.5.1 Comparison of MTLBO with basic TLBO

To validate the performance of MTLBO, the experimental outcomes and comparison of the proposed algorithm with

Table 8 Comparison of MTLBO algorithm on Carlier benchamark datasets

Problem	Size	Z_{opt}		DE	PSOVNS	DBA	HBSA	PSOMA	MTLBO
Car01	11×5	7038	BRPD	0	0	0	0	0	**0**
			ARPD	0	0	0	0	0	**0**
			WRPD	0	0	0	0	0	**0**
Car02	13×4	7166	BRPD	0	0	0	0	0	**0**
			ARPD	0	0	0	0	0	**0**
			WRPD	0	0	0	0	0	**0**
Car03	12×5	7312	BRPD	0	0	0	0	0	**0**
			ARPD	0.0369	0.42	0.397	0.06	0	**0**
			WRPD	0.7385	1.189	1.19	1.19	0	**0**
Car04	14×4	8003	BRPD	0	0	0	0	0	**0**
			ARPD	0	0	0	0	0	**0**
			WRPD	0	0	0	0	0	**0**
Car05	10×6	7720	BRPD	0	0	0	0	0	**0**
			ARPD	0.5285	0.039	0	0	0.018	**0**
			WRPD	1.3083	0.389	0	0	0.375	**0**
Car06	8×9	8505	BRPD	0	0	0	0	0	**0**
			ARPD	0.3821	0.076	0	0	0.114	**0**
			WRPD	0.7643	0.764	0	0	0.764	**0**
Car07	7×7	6590	BRPD	0	0	0	0	0	**0**
			ARPD	0.5341	0	0	0	0	**0**
			WRPD	2.868	0	0	0	0	**0**
Car08	8×8	8366	BRPD	0	0	0	0	0	**0**
			ARPD	0.0645	0	0	0	0	**0**
			WRPD	0.6455	0	0	0	0	**0**

basic TLBO for Taillard benchmark test functions are shown in Table 11. The performance measure indicators such as *BRPD*, *ARPD* and *WRPD* are given in Table 11.

MTLBO algorithm outperforms the basic TLBO algorithm on all Taillard benchmark functions. The total average values for *ARPD*, *BRPD* and *WRPD* of MTLBO are 0.830, 3.717 and 6.118 which are better than basic TLBO algorithm tested values.

The convergence plots of basic TLBO and MTLBO algorithms are listed in Fig. 11. For plotting the graph, makespan versus Taillard benchmark function (such as TA010, TA030, TA050, TA070, TA090 and TA0110) are used. From these graphs, it can be indicated that the proposed algorithm converges faster than basic TLBO algorithm. As the iterations increase, MTLBO hardly finds accurate solutions, but by embedding the local search operator into MTLBO, it can achieve better performance.

5.5.2 Comparison of MTLBO with other EAs

In this subsection, the proposed MTLBO algorithm is compared with IG1 [77], IG2 [77], HDDE [78] and TLBO algorithms using Taillard benchmark test functions. The performance is measured using *ARPD* indicator. Table 12 shows comparison of proposed algorithm over population based algorithms with respect to ARPD. The MTLBO algorithm obtained lowest average value i.e. 0.83 for ARPD, which is better than IG1 (2.31), IG2 (2.27), HDDE (4.51) and TLBO (3.11).

Later on, the proposed MTLBO algorithm is compared with GA [79], HPSO [78], ATPPSO [80] and NCS [81] on the Taillard benchmark test functions. The computation results achieved by GA, HPSO, ATPPSO and NCS are indicated in Table 13. The performance measure indicators such as Mean and *ARPD* (Average relative error) are also

Table 9 Comparison of MTLBO algorithm on Reeves benchmark datasets

Problem	Size	Z_{opt}		DE	PSOVNS	DBA	PSOMA	HBSA	HCS	MTLBO
REC001	20×5	1247	BRPD	0	0.16	0	0	0	0	**0**
			ARPD	0	0.168	0.08	0.144	0.14	0.016	**0**
			WRPD	0	0.321	0.16	0.16	0.16	0.1602	**0**
REC003	20×5	1109	BRPD	0	0	0	0	0	0	**0**
			ARPD	0	0.158	0.081	0.189	0.08	0	**0**
			WRPD	0	0.18	0.18	0.721	0.18	0	**0**
REC005	20×5	1242	BRRD	0.2415	0.242	0.241	0.242	0.24	0.2415	**0**
			ARPD	0.2415	0.249	0.241	0.249	0.24	0.2415	**0**
			WRPD	0.2415	0.42	0.241	0.402	0.24	0.2415	**0**
REC007	20×10	1566	BRPD	0	0.702	0	0	0	0	**0**
			ARPD	0	1.095	0.575	0.986	0.46	0	**0**
			WRPD	0	1.405	1.149	1.149	1.15	0	**0**
REC009	20×10	1537	BRPD	0	0	0	0	0	0	**0**
			ARPD	0.026	0.651	0.638	0.621	0.07	0	**0**
			WRPD	0.026	1.366	2.407	1.691	0.65	0	**0**
REC011	20×10	1431	BRPD	0	0.071	0	0	0	0	**0**
			ARPD	0	1.153	1.167	0.129	0	0	**0**
			WRPD	0	2.656	2.655	0.978	0	0	**0**
REC013	20×15	1930	BRPD	0	1.036	0.415	0.259	0.1	0	**0**
			ARPD	0.2746	1.79	1.461	0.893	0.53	0.1917	**0**
			WRPD	0.7772	2.643	3.782	1.502	1.14	0.3109	**0**
REC015	20×15	1950	BRPD	0	0.769	0.154	0.051	0.05	0	**0**
			ARPD	0.5231	1.487	1.226	0.628	0.64	0.1128	**0**
			WRPD	1.1795	2.256	2.103	1.076	1.18	0.4615	**0**
REC017	20×15	1902	BRPD	0	0.999	0.368	0	0	0	**0**
			ARPd	0.3628	2.453	1.277	1.33	1	0.1157	**0**
			WRPD	0.9464	3.365	2.154	2.155	2.16	0.4732	**0**
REC019	30×10	2093	BRPD	0.2867	1.529	0.573	0.43	0.29	0.1433	**0**
			ARPD	0.7023	2.099	0.929	1.313	0.81	0.5638	**0.247**
			WRPD	1.2422	2.532	2.023	2.102	1.29	0.7645	**0.282**
REC021	30×10	2017	BRPD	0.6445	1.487	1.438	1.437	0.69	0.2975	**0**
			ARPD	1.2791	1.671	1.671	1.596	1.5	1.1651	**0.112**
			WRPD	1.4378	2.033	2.231	1.636	1.83	1.4378	**0.109**
REC023	30×10	2011	BRPD	0.3481	1.343	0.796	0.596	0.45	0.248	**0**
			ARPD	0.4276	2.106	1.173	1.31	1.28	0.4625	**0.202**
			WRPD	0.4973	2.884	2.381	2.038	3.08	0.4973	**0.317**
REC025	30×15	2513	BRPD	0.5571	2.388	1.632	0.835	0.4	0.2786	**0**
			ARPD	1.0824	0.16	2.921	2.085	1.29	0.8754	**0.034**
			WRPD	1.6315	0.168	3.94	3.233	2.43	0.154	**0.122**
REC027	30×15	2373	BRPD	0.2528	1.728	1.011	1.348	0.25	0.2528	**0**
			ARPD	0.8512	2.463	1.419	1.605	1.27	0.8175	**0.061**
			WRPD	1.2221	3.203	2.298	2.402	2.57	1.0535	**0.121**
REC029	30×15	2287	BRPD	0.8308	1.968	1.049	1.442	0.57	0.0875	**0**
			ARPD	1.0494	3.109	2.58	1.888	1.42	0.8833	**0**
			WRPD	1.443	4.067	3.935	2.492	2.97	1.1369	**0**
REC031	50×10	3045	BRPD	0.427	2.594	2.299	1.51	0.43	0.263	**0.214**
			ARPD	0.644	3.232	3.392	2.254	1.91	0.8998	**0.202**
			WRPD	0.92	4.237	4.532	2.692	2.66	1.3793	**0.214**
REC033	50×10	3114	BRPD	0	0.835	0.61	0	0	0	**0**
			ARPD	0.244	1.007	0.728	0.645	0.59	0.5652	**0**
			WRPD	0.835	1.477	1.734	0.834	1.28	0.8349	**0**

Table 9 (continued)

Problem	Size	Z_{opt}		DE	PSOVNS	DBA	PSOMA	HBSA	HCS	MTLBO
REC035	50 × 10	3277	BRPD	0	0	0	0	0	0	**0**
			ARPD	0	0.038	0.037	0	0	0	**0**
			WRPD	0	0.092	0.092	0	0	0	**0**
REC037	75 × 20	4951	BRPD	2.565	4.383	3.373	2.101	1.92	1.636	**1.643**
			ARPD	3.001	4.949	4.872	3.537	2.93	2.2541	**1.852**
			WRPD	3.555	5.736	5.979	4.039	4.2	2.5045	**2.286**
REC039	75 × 20	5087	BRPD	1.73	2.85	2.28	1.553	0.9	0.806	**0.654**
			ARPD	1.832	3.371	3.851	2.426	1.88	1.2463	**0.811**
			WRPD	2.005	3.951	5.347	2.83	3.38	1.553	**0.914**
REC041	75 × 20	4960	BRPD	2.661	4.173	3.81	2.641	1.69	1.4516	**0.986**
			ARPD	3.35	4.867	5.095	3.684	2.72	2.1532	**1.684**
			WRPD	3.77	5.585	6.532	4.052	3.55	2.4597	**2.451**

Table 10 Comparison of *MTLBO* algorithm on Heller benchmark datasets

Problems	Size	Max bound	NEH	DSOMA	SG	*MTLBO*
Hel1	20 × 10	516	518	631	515	**515**
Hel2	10 × 10	136	141	150	137	**135**

Table 11 Results achieved by MTLBO and TLBO for Taillard benchmark datasets

Problems	Size	TLBO			MTLBO		
		ARE	BRE	WRE	ARE	BRE	WRE
TA001–TA010	20 × 5	0.780	1.976	4.987	**0.000**	**1.314**	**4.786**
TA010–TA020	20 × 10	1.520	2.996	4.345	**0.880**	**2.964**	**3.987**
TA020–TA030	20 × 20	2.080	5.231	10.654	**0.240**	**4.987**	**9.886**
TA030–TA040	50 × 5	0.200	2.123	5.321	**0.000**	**2.077**	**4.342**
TA040–TA050	50 × 10	5.030	3.802	8.098	**1.920**	**3.786**	**7.674**
TA050–TA060	50 × 20	6.860	3.621	7.210	**2.830**	**3.564**	**6.987**
TA060–TA070	100 × 5	0.750	1.090	1.978	**0.060**	**0.870**	**1.046**
TA070–TA080	100 × 10	1.570	4.099	6.045	**0.620**	**3.990**	**5.876**
TA080–TA090	100 × 20	5.960	8.099	9.065	**2.210**	**7.786**	**8.954**
TA090–TA100	200 × 10	4.780	2.188	5.023	**0.520**	**2.098**	**4.988**
TA100–TA110	200 × 20	4.080	6.791	8.102	**3.060**	**6.596**	**7.987**
TA110–TA120	500 × 20	3.720	4.678	7.105	**1.660**	**4.567**	**6.909**
Average		2.738	3.891	6.494	**0.830**	**3.717**	**6.118**

presented in Table 13. The best outcomes of the benchmark functions are indicated in bold.

In Table 13, MTLBO obtains the best computational outcomes with respect to all the given metaheuristic algorithms. The MTLBO algorithm obtained lowest total mean *ARPD* (i.e. 0.8) which is superior value than HPSO (2.74), ATPPSO (1.94), GA (4.31), and NCS (1.24). The performance of proposed algorithm significantly superior outcomes over Taillard benchmark test function. The mean values of proposed algorithm are within the lower and upper bound given as OR-Library.

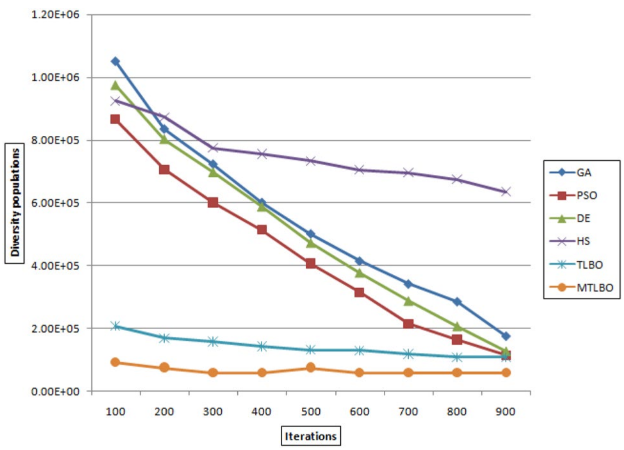


Fig. 7 Convergence graph of population diversity of unimodal function (fun2: Schwefels) for metaheuristic algorithms (D = 50)

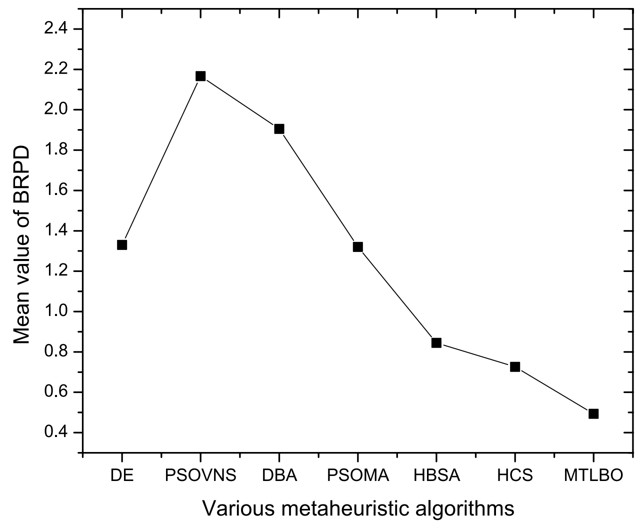


Fig. 8 Comparison of mean value of *BRPD* on Reeves benchmark datasets

In Table 14, the proposed algorithm compared total flowshop time criterion over Taillard benchmark problems which taken from http://www.cs.colostate.edu/sched/generator. MTLBO compared with PSO and RSA [82] . The mimumium and average results obtained by MTLBO is significantly superior than PSO and RSA.

To analyze the performance of MTLBO algorithm over other evolutionary algorithms, Wilcoxon sign ranked test was conducted by using achieved *ARPD* value. Table 15 shows significant *p*-value of proposed algorithm over other evolutionary algorithms. From these experimental outcomes, it is observed that MTLBO algorithm is more effective and significant than GA, HPSO, ATPPSO and NCS. It is to be noted that the *p*-value in this experiment is shown in bold letters for below 0.05 value.

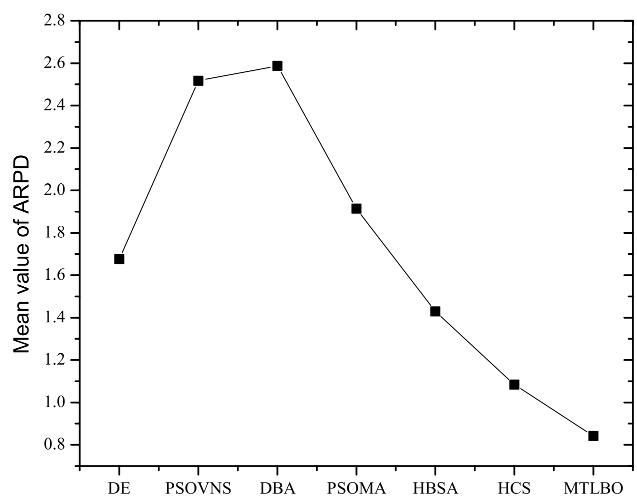


Fig. 9 Comparison of mean value of *ARPD* on Reeves benchmark datasets

Table 12 Comparison of MTLBO with other population based algorithms on basis of ARPD using Taillard Benchmark datasets

Problem	PS(J X M)	IG1	IG2	HDDE	TLBO	MTLBO
Ta010	20 × 05	0.39	0.46	1.49	0.78	**0**
Ta020	20 × 10	0.48	0.62	1.53	1.52	**0.88**
Ta030	20 × 20	0.31	0.32	1.23	2.08	**0.24**
Ta040	50 × 05	2.71	2.99	5.69	0.20	**0**
Ta050	50 × 10	3.24	3.23	5.63	5.03	**1.92**
Ta060	50 × 20	2.88	2.54	5.04	6.86	**2.83**
Ta070	100 × 05	3.82	3.56	7.22	0.75	**0.06**
Ta080	100 × 10	3.34	3.48	6.67	1.57	**0.62**
Ta090	100 × 20	3.03	2.82	4.41	5.96	**2.21**
Ta0100	200 × 10	3.85	3.63	6.91	4.78	**0.52**
Ta0110	200 × 20	2.31	2.20	4.34	4.08	**3.06**
Ta0120	500 × 20	1.32	1.33	3.93	3.72	**1.66**
Average		2.31	2.27	4.51	3.11	**0.83**

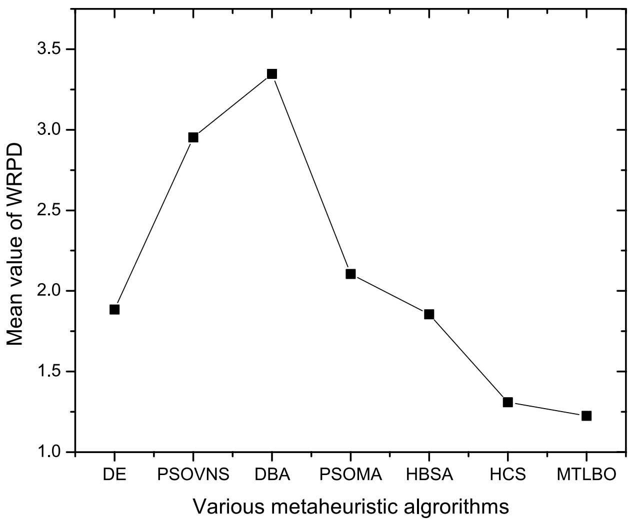


Fig. 10 Comparison of mean value of *WRPD* on Reeves benchmark datasets

5.6 MTLBO algorithm on VRF benchmark datasets

New benchmark test functions such as VRF_hard_small benchmark and VRF_hard_large benchmark are proposed in [70]. These benchmark functions are selected to perform experiments on PFSSP. These functions consist of 240 small instances and 240 large instances, with upto 800 jobs on 60 machines. The small instances have following combinations of n = number of jobs, m = number of machines i.e. n = [10, 20, 30, 40, 50, 60] and m = [5, 10, 15, 20] and large instances consist of n = [100, 200, 300, 400, 500, 600, 700, 800] and m = [20, 40, 60].

The stopping criterion to perform VRF benchmark function is evaluated with maximum running CPU time as follows

$$Maximum\ running\ CPU\ time = (n.(m/2).t) \tag{20}$$

where t indicates the amount of time (t = 5 ms). Each of the instance runs at least 25 times to obtain the best outcomes.

To measure the effectiveness of the metaheuristic method, *ARPD* is calculated for each instance as in Eq. (17).

The computational results of the MTLBO over different metaheursitic algorithms with VRF small instance are given in the Table 16. The best outcomes of the experimental are shown in bold letters. The performance of MTLBO is calculated using *ARPD* and *SD* values. In Table 16, the average value for MTLBO with VRF small instances for *ARPD* and *SD* are 0.10 and 0.03 respectively. From Table 16, it is observed that *ARPD* and *SD* value of MTLBO is superior than DPSO, GA-VNS, HDE and HPSO algorithms.

The computational results of MTLBO over different metaheursitic algorithms with VRF Large instances are given in the Table 17. The best outcomes of the

Table 13 Comparison of MTLBO algorithm with other evolutionary algorithms over Taillard benchmark problems

Problems	Size	GA		HPSO		ATPPSO		NCS		MTLBO	
		ARPD	Mean	ARPD	Mean	ARPD	Mean	ARPD	Mean	ARPD	Mean
Ta010	20 × 05	2.49	1135.6	0.68	1278	0.22	1110.4	0.00	1108	**0**	**1108**
Ta020	20 × 10	2.63	1632.8	1.94	1587.5	1.09	1608.3	0.94	1606	**0.9**	**1598**
Ta030	20 × 20	2.73	2237.5	1.44	2307	0.72	2193.6	0.28	2184	**0.2**	**2180**
Ta040	50 × 05	1.21	2815.7	0.08	2724	0.02	2782.5	0.00	2782	**0**	**2782**
Ta050	50 × 10	5.24	3225.5	3.09	3053.6	2.97	3156.1	2.16	3131.2	**1.9**	**3085**
Ta060	50 × 20	6.73	4008.6	5.46	3944.6	3.92	3903.2	2.78	3860.6	**2.8**	**3752**
Ta070	100 × 05	0.95	5372.3	0.75	5493	0.43	5344.9	0.08	5326	**0.1**	**5324**
Ta080	100 × 10	3.62	6056.3	1.57	5913.3	0.94	5900.1	0.79	5891.4	**0.6**	**5844**
Ta090	100 × 20	7.40	6910.3	5.86	6598.5	3.99	6690.6	2.62	6602.8	**2.2**	**6430**
Ta0100	200 × 10	3.28	11025.6	4.88	10796.9	1.60	10846.2	0.55	10734	**0.5**	**10675**
Ta0110	200 × 20	8.70	12269.5	3.98	11,832.1	4.39	11,783	3.06	11,633.6	**3.1**	**11,284**
Ta0120	500 × 20	6.76	28,245	3.12	27,282	2.98	27,246.5	1.66	26,897.2	**1.7**	**26,453**
Average		4.31		2.74		1.94		1.24		**0.8**	

Table 14 Comparision of MTLBO on total flowshop time criterion over Taillard benchmark problems

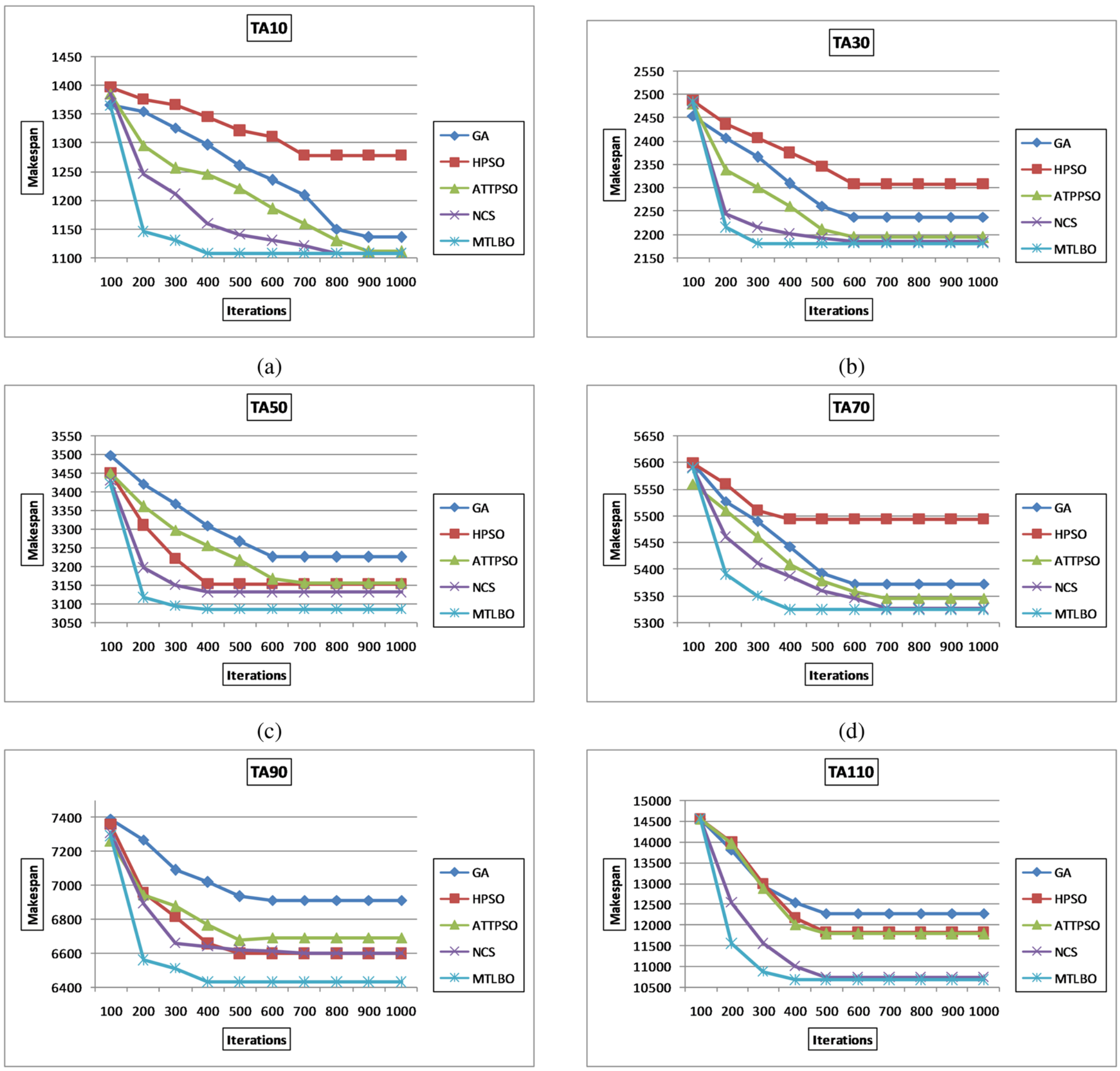
Problems	PSO		RSA		MTLBO	
	MIN	Average	MIN	Average	MIN	Average
20 × 20	34,518	35,026.50	32,443	33,770.60	32,273	33,570.40
20 × 20	33,243	34,039.80	32,166	33,406.90	31,892	33,362.20
20 × 20	35,489	36,207.80	35,066	35,610.70	34,752	35,610.50
20 × 20	32,234	32,836.50	30,915	31,925.50	30,112	31,621.50
20 × 20	35,086	35,471.40	34,354	35,027.40	33,786	34,786.80
50 × 20	136,668	138,212.60	139,522	142,179.60	138,986	142,111.20
50 × 20	136,262	140,163.70	141,745	142,899	141,672	141,765
50 × 20	138,031	142,076.30	145,079	146,329.20	144,872	145,984.80
50 × 20	134,550	137,505.70	139,607	140,680.70	138,211	140,113.70
50 × 20	135,915	138,541.90	140,990	142,256	134,651	141,986
100 × 20	428,602	433,439.20	428,602	433,439.20	428,032	433,112.90
100 × 20	430,677	434,747.60	430,677	434,747.60	430,654	433,211.90
100 × 20	417,704	426,910.40	417,704	426,910.40	417,702	425,872.40
100 × 20	418,411	427,356.90	418,411	427,356.90	418,409	426,112.90
100 × 20	416,454	424,499.50	416,454	424,499.50	416,454	423,212.20
200 × 20	1,421,104	1,440,869	1,421,104	1,440,869	1,421,104	1,440,762
200 × 20	1,447,880	1,463,356	1,447,880	1,463,356	1,447,878	1,463,119
200 × 20	1,462,239	1,479,747	1,462,239	1,479,747	1,462,239	1,479,117
200 × 20	1,434,704	1,450,640	1,434,704	1,450,640	1,434,702	1,450,081
200 × 20	1,434,687	1,456,151	1,434,687	1,456,151	1,434,687	1,455,113

experiments is shown in bold letters. The performance of MTLBO algorithm is calculated using *ARPD* and *SD* values. In Table 17, the average value of the MTLBO with VRF small instances for *ARPD* and *SD* are 2.61 and 0.14 respectively. From Table 17, it is observed that obtained results using *ARPD* and *SD* value for MTLBO algorithm are statistically better than DPSO, GA-VNS, HDE and HPSO algorithms.

6 Conclusion

This paper proposed a Modified Teaching Learning Based Optimization with Opposite Based Learning approach to find solution of Permutation Flow-Shop Scheduling Problem. OBL approach is used to enhance the quality of the initial population and convergence speed. MTLBO is developed to determine the PFSSP efficiently using the Largest Order Value rule-based random-key, to change an individual into discrete job schedules. Based on Nawaz–Enscore–Ham heuristic mechanism, the new initial populations are generated in MTLBO. To enhance the local exploitation ability, the effective swap, insert and inverse structures are incorporated into the MTLBO.

The statistical result and analysis of the MTLBO based on ten benchmark functions and Wilcoxon signed rank test shows the effectiveness of the algorithm. The computational results over five well-known benchmark functions such as Carlier, Reeves, Heller, Taillard and VRF benchmark instances set indicates that MTLBO is the most powerful and convenient method to solve PFSSP. The performance measures of the proposed algorithm are calculated using *ARPD*,



(a) (b) (c) (d) (e) (f)

Fig. 11 Convergence plot of makespan over MTBLO and metaheuristic algorithms on Taillard functions

Table 15 The p-value obtained by MTLBO versus other metaheuristic algorithms ($p \leq 0.05$)

Metaheuristic algorithms	p-value
MTLBO versus HPSO	**3.224E−03**
MTLBO versus GA	**3.220E−03**
MTLBO versus ATPPSO	**8.675E−04**
MTLBO versus NCS	**2.210E−03**

BRPD, and *WRPD* over all datasets. The performance of MTLBO is tested against state-of-the-art algorithms. It is observed that MTLBO algorithm works effectively in most of the cases. Thus, by conducting Wilcoxon signed test, it is proved that MTLBO algorithm is much more effective than other metaheuristic algorithms. The future research direction focus on hybrid flow shop scheduling and multi-objective flow shop problems.

Table 16 Comparison of MTLBO algorithm with other evolutionary algorithms over VRF hard small benchmark (t = 5 ms)

Instances	DPSO		GA-VNS		HDE		HPSO		MTLBO	
	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
10 × 05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**
10 × 10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**
10 × 15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**
10 × 20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**
20 × 05	0.02	0.03	0.01	0.02	0.04	0.05	0.01	0.03	**0.00**	**0.00**
20 × 10	0.00	0.00	0.00	0.00	0.00	0.00	0.38	0.18	**0.00**	**0.00**
20 × 15	0.01	0.01	0.00	0.00	0.00	0.00	0.40	0.30	**0.00**	**0.00**
20 × 20	0.00	0.00	0.00	0.00	0.00	0.00	0.28	0.22	**0.00**	**0.00**
30 × 05	0.18	0.14	0.25	0.15	0.34	0.21	0.17	0.22	**0.07**	**0.04**
30 × 10	0.14	0.15	0.09	0.08	0.18	0.24	0.42	0.41	**0.00**	**0.00**
30 × 15	0.11	0.13	0.01	0.01	0.11	0.10	0.32	0.37	**0.01**	**0.01**
30 × 20	0.17	0.12	0.02	0.01	0.13	0.09	1.02	0.32	**0.00**	**0.00**
40 × 05	0.53	0.21	0.87	0.26	1.00	0.28	0.09	0.32	**0.22**	**0.03**
40 × 10	0.31	0.22	0.39	0.18	0.48	0.22	2.18	0.40	**0.06**	**0.04**
40 × 15	0.21	0.19	0.24	0.14	0.25	0.16	2.02	0.48	**0.00**	**0.00**
40 × 20	0.22	0.13	0.18	0.11	0.25	0.15	1.76	0.47	**0.00**	**0.00**
50 × 5	0.69	0.23	0.39	0.26	1.36	0.36	1.74	0.38	**0.32**	**0.11**
50 × 10	0.44	0.19	0.76	0.24	0.67	0.26	2.57	0.40	**0.16**	**0.05**
50 × 15	0.36	0.21	0.69	0.21	0.58	0.25	2.38	0.51	**0.08**	**0.03**
50 × 20	0.34	0.16	0.65	0.25	0.58	0.19	2.39	0.49	**0.00**	**0.00**
60 × 05	1.02	0.26	2.11	0.33	1.75	0.31	2.37	0.49	**0.50**	**0.15**
60 × 10	0.71	0.25	1.37	0.25	1.08	0.25	2.97	0.39	**0.26**	**0.12**
60 × 15	0.52	0.25	1.34	0.26	0.95	0.24	3.00	0.57	**0.45**	**0.08**
60 × 20	0.51	0.22	1.07	0.28	0.89	0.27	3.07	0.47	**0.23**	**0.04**
Average	0.27	0.13	0.44	0.13	0.44	0.15	1.23	0.31	**0.10**	**0.03**

Table 17 Comparison of MTLBO algorithm with other evolutionary algorithms over VRF hard large benchmark (t = 5 ms)

Instances	DPSO		GA-VNS		HDE		HPSO		MTLBO	
	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
100 × 20	1.09	0.25	0.89	0.20	1.86	0.23	4.98	0.43	**0.51**	**0.07**
100 × 40	0.95	0.28	0.75	0.31	1.60	0.20	4.97	0.45	**0.32**	**0.14**
100 × 60	0.84	0.25	0.74	0.26	1.43	0.18	4.28	0.47	**0.72**	**0.24**
200 × 20	2.38	0.21	2.45	0.33	3.90	0.26	4.95	0.42	**1.05**	**0.14**
200 × 40	2.36	0.31	2.05	0.30	3.41	0.25	4.92	0.43	**0.93**	**0.10**
200 × 60	1.84	0.24	1.75	0.30	3.24	0.24	5.13	0.62	**0.79**	**0.12**
300 × 20	3.18	0.28	3.27	0.25	4.54	0.36	5.53	0.64	**1.51**	**0.14**
300 × 40	2.81	0.25	2.82	0.24	4.16	0.37	5.67	0.60	**1.56**	**0.13**
300 × 60	2.61	0.21	2.51	0.28	4.11	0.43	5.36	0.65	**1.78**	**0.20**
400 × 20	3.88	0.25	4.00	0.28	5.02	0.78	5.95	0.63	**1.87**	**0.13**
400 × 40	3.43	0.33	3.33	0.18	4.60	0.54	6.97	0.72	**2.76**	**0.12**
400 × 60	3.24	0.20	4.78	0.29	4.87	0.58	6.95	0.73	**1.97**	**0.11**
500 × 20	4.60	0.24	3.98	0.97	4.76	0.52	6.54	0.71	**2.03**	**0.15**
500 × 40	4.08	0.28	3.09	0.37	5.87	0.64	6.43	0.74	**5.89**	**0.15**
500 × 60	3.82	0.24	3.87	0.67	5.12	0.86	7.86	0.79	**2.17**	**0.13**
600 × 20	5.21	0.26	5.98	0.54	5.67	0.83	7.96	0.81	**2.76**	**0.14**
600 × 40	4.90	0.26	4.76	0.87	4.83	0.63	7.65	0.79	**5.15**	**0.19**
600 × 60	4.60	0.27	4.87	0.26	6.45	0.92	7.78	0.82	**3.76**	**0.14**
700 × 20	6.60	0.32	4.76	0.56	5.65	0.65	8.56	0.89	**3.08**	**0.15**
700 × 40	5.54	0.22	5.76	0.87	4.87	0.52	8.96	0.88	**5.79**	**0.15**
700 × 60	5.49	0.30	4.88	0.45	4.56	0.48	9.76	0.94	**4.89**	**0.19**
800 × 20	6.88	0.27	5.67	0.27	6.35	0.84	9.98	0.94	**3.87**	**0.14**
800 × 40	6.37	0.25	4.87	0.76	6.76	0.96	9.65	0.97	**3.98**	**0.15**
800 × 60	6.06	0.22	4.39	0.65	4.87	0.63	7.87	0.94	**3.56**	**0.10**
Average	3.87	0.26	3.59	0.44	4.52	0.54	6.86	0.71	**2.61**	**0.14**

References

1. Pinedo ML (2016) Scheduling: theory, algorithms, and systems. Springer, Berlin. https://doi.org/10.1007/978-1-4614-2361-4
2. Grabowski J, Pempera J (2000) Sequencing of jobs in some production system. Eur J Oper Res 125(3):535–550. https://doi.org/10.1016/S0377-2217(99)00224-6
3. Du W, Tang Y, Leung SYS, Tong L, Vasilakos AV, Qian F (2018) Robust order scheduling in the discrete manufacturing industry: a multiobjective optimization approach. IEEE Trans Ind Inf 14(1):253–264. https://doi.org/10.1109/TII.2017.2664080
4. Hidri L, Gharbi A (2017) New efficient lower bound for the hybrid flow shop scheduling problem with multiprocessor tasks. IEEE Access 5:6121–6133. https://doi.org/10.1109/ACCESS.2017.2696118
5. Bargaoui H, Driss OB, Ghédira K (2017) Towards a distributed implementation of chemical reaction optimization for the multi-factory permutation flowshop scheduling problem. Procedia Comput Sci 112:1531–1541. https://doi.org/10.1016/j.procs.2017.08.057
6. Deng F, He Y, Zhou S, Yu Y, Cheng H, Wu X (2018) Compressive strength prediction of recycled concrete based on deep learning. Constr Build Mater 175:562–569
7. Cho HM, Jeong IJ (2017) A two-level method of production planning and scheduling for bi-objective reentrant hybrid flow shops. Comput Ind Eng 106:174–181. https://doi.org/10.1016/j.cie.2017.02.010
8. Pan QK (2016) An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. Eur J Oper Res 250(3):702–714. https://doi.org/10.1016/j.ejor.2015.10.007
9. Zhang Y, Wang J, Liu Y (2017) Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact. J Clean Prod 167:665–679. https://doi.org/10.1016/j.jclepro.2017.08.068
10. Johnson SM (1954) Optimal two and three stage production schedules with setup times included. Naval Res Logist Q 1(1):61–68. https://doi.org/10.1002/nav.3800010110
11. Ruiz R, Stützle T (2008) An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur J Oper Res 187(3):1143–1159. https://doi.org/10.1016/j.ejor.2006.07.029
12. Ozolins A (2017) Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. Central Eur J Oper Res. https://doi.org/10.1007/s10100-017-0488-5
13. Toumi S, Jarboui B, Eddaly M, Rebaï A (2017) Branch-and-bound algorithm for solving blocking flowshop scheduling problems with makespan criterion. Int J Math Oper Res 10(1):34–48. https://doi.org/10.1504/IJMOR.2017.080743
14. Selen WJ, Hott DD (1986) A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem. J Oper Res Soc 37(12):1121–1128. https://doi.org/10.1057/jors.1986.197
15. Grabowski J, Wodecki M (2004) A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Comput Oper Res 31(11):1891–1909. https://doi.org/10.1016/S0305-0548(03)00145-X
16. Xu J, Zhang J (2014) Exploration-exploitation tradeoffs in metaheuristics: survey and analysis. In: Proceedings of the 33rd Chinese control conference, pp 8633–8638. https://doi.org/10.1109/ChiCC.2014.6896450
17. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99. https://doi.org/10.1023/A:1022602019183
18. Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theoret Comput Sci 344(2):243–278. https://doi.org/10.1016/j.tcs.2005.05.020
19. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359. https://doi.org/10.1023/A:1008202821328
20. Geem ZW, Kim JH, Loganathan G (2001) A new heuristic optimization algorithm: Harmony search. SIMULATION 76(2):60–68. https://doi.org/10.1177/003754970107600201
21. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, Springer-Verlag, Berlin, Heidelberg, SAGA'09, pp 169–178
22. Yang XS (2010) A new metaheuristic bat-inspired algorithm. Springer, Berlin, pp 65–74
23. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35. https://doi.org/10.1007/s00366-011-0241-y
24. Rao R, Savsani V, Vakharia D (2011) Teaching learning based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315. https://doi.org/10.1016/j.cad.2010.12.015
25. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
26. Wang GG, Deb S, Coelho L (2015a) Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Int J Bioinspir Comput 12:1–22
27. Wang GG, Deb S, Cui Z (2015b) Monarch butterfly optimization. Neural Comput Appl. https://doi.org/10.1007/s00521-015-1923-y
28. Karaboga D, Basturk B (2007) Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In: Foundations of fuzzy logic and soft computing. Springer, Berlin, pp 789–798
29. Rao R (2016) Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput 7(1):19–34
30. Wang GG (2016) Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memet Comput. https://doi.org/10.1007/s12293-016-0212-3
31. Zhang Y, fang Song X, wei Gong D, (2017) A return-cost-based binary firefly algorithm for feature selection. Inf Sci 418–419:561–574. https://doi.org/10.1016/j.ins.2017.08.047
32. Reeves CR (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22(1):5–13. https://doi.org/10.1016/0305-0548(93)E0014-K, genetic Algorithms
33. Mirabi M, Fatemi Ghomi SMT, Jolai F (2014) A novel hybrid genetic algorithm to solve the make-to-order sequence-dependent flow-shop scheduling problem. J Ind Eng Int 10(2):57. https://doi.org/10.1007/s40092-014-0057-7
34. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177(3):1930–1947. https://doi.org/10.1016/j.ejor.2005.12.024
35. Liu D, Tan KC, Goh CK, Ho WK (2007) A multiobjective memetic algorithm based on particle swarm optimization. IEEE Trans Syst Man Cybern Part B (Cybern) 37(1):42–50. https://doi.org/10.1109/TSMCB.2006.883270
36. Gao J, Chen R, Deng W (2013) An efficient tabu search algorithm for the distributed permutation flowshop

scheduling problem. Int J Prod Res 51(3):641–651. https://doi.org/10.1080/00207543.2011.644819
37. Liu Y, Yin M, Gu W (2014) An effective differential evolution algorithm for permutation flow shop scheduling problem. Appl Math Comput 248:143–159. https://doi.org/10.1016/j.amc.2014.09.010
38. Li X, Yin M (2013) A hybrid cuckoo search via lévy flights for the permutation flow shop scheduling problem. Int J Prod Res 51(16):4732–4754. https://doi.org/10.1080/00207543.2013.767988
39. Sioud A, Gagné C (2018) Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. Eur J Oper Res 264(1):66–73. https://doi.org/10.1016/j.ejor.2017.06.027
40. Fernandez-Viagas V, Perez-Gonzalez P, Framinan JM (2018) The distributed permutation flow shop to minimise the total flowtime. Comput Ind Eng 118:464–477. https://doi.org/10.1016/j.cie.2018.03.014
41. Meng T, Pan QK, Li JQ, Sang HY (2018) An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem. Swarm Evol Comput 38:64–78. https://doi.org/10.1016/j.swevo.2017.06.003
42. Han Y, Gong D, Li J, Zhang Y (2016) Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int J Prod Res 54(22):6782–6797. https://doi.org/10.1080/00207543.2016.1177671
43. Baykasolu A, Hamzadayi A, Köse SY (2014) Testing the performance of teachinglearning based optimization (tlbo) algorithm on combinatorial problems: flow shop and job shop scheduling cases. Inf Sci 276:204–218. https://doi.org/10.1016/j.ins.2014.02.056
44. Xie Z, Zhang C, Shao X, Lin W, Zhu H (2014) An effective hybrid teachinglearning-based optimization algorithm for permutation flow shop scheduling problem. Adv Eng Softw 77:35–47. https://doi.org/10.1016/j.advengsoft.2014.07.006
45. Shao W, Pi D, Shao Z (2016) A hybrid discrete optimization algorithm based on teachingprobabilistic learning mechanism for no-wait flow shop scheduling. Knowl Based Syst 107:219–234. https://doi.org/10.1016/j.knosys.2016.06.011
46. qing Li J, ke Pan Q, Mao K, (2015) A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. Eng Appl Artif Intell 37:279–292. https://doi.org/10.1016/j.engappai.2014.09.015
47. Shao W, Pi D, Shao Z (2017) An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem. Appl Soft Comput 61:193–210. https://doi.org/10.1016/j.asoc.2017.08.020
48. Buddala R, Mahapatra SS (2017) Improved teaching-learning-based and jaya optimization algorithms for solving flexible flow shop scheduling problems. J Ind Eng Int 1:1. https://doi.org/10.1007/s40092-017-0244-4
49. Abdel-Basset M, Manogaran G, El-Shahat D, Mirjalili S (2018) A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Future Gener Comput Syst 85:129–145. https://doi.org/10.1016/j.future.2018.03.020
50. Rao RV, Patel V (2013) Multi-objective optimization of two stage thermoelectric cooler using a modified teachinglearning-based optimization algorithm. Eng Appl Artif Intell 26(1):430–445. https://doi.org/10.1016/j.engappai.2012.02.016
51. Satapathy SC, Naik A (2011) Data clustering based on teaching-learning-based optimization. In: Panigrahi BK, Suganthan PN, Das S, Satapathy SC (eds) Swarm, evolutionary, and memetic computing. Springer, Heidelberg, pp 148–156
52. Jin H, Wang Y (2014) A fusion method for visible and infrared images based on contrast pyramid with teaching learning based optimization. Infrared Phys Technol 64:134–142. https://doi.org/10.1016/j.infrared.2014.02.013
53. Rao RV, Patel V (2013) An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. Sci Iran 20(3):710–720. https://doi.org/10.1016/j.scient.2012.12.005
54. Waghmare G (2013) Comments on a note on teachinglearning-based optimization algorithm. Inf Sci 229:159–169. https://doi.org/10.1016/j.ins.2012.11.009
55. Das S, Konar A, Chakraborty UK (2005) Two improved differential evolution schemes for faster global search. In: Proceedings of the 7th annual conference on genetic and evolutionary computation. GECCO ’05. ACM, New York, pp 991–998. https://doi.org/10.1145/1068009.1068177
56. Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160. https://doi.org/10.1287/ijoc.6.2.154
57. Li X, Yin M (2013) An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. Adv Eng Softw 55:10–31. https://doi.org/10.1016/j.advengsoft.2012.09.003
58. Qian B, Wang L, Hu R, Wang WL, Huang DX, Wang X (2008) A hybrid differential evolution method for permutation flow-shop scheduling. Int J Adv Manuf Technol 38(7):757–777. https://doi.org/10.1007/s00170-007-1115-8
59. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC’06), vol 1, pp 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
60. Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79. https://doi.org/10.1109/TEVC.2007.894200
61. Wang H, Wu Z, Rahnamayan S, Kang L (2009) A scalability test for accelerated de using generalized opposition-based learning. In: 2009 ninth international conference on intelligent systems design and applications, pp 1090–1095. https://doi.org/10.1109/ISDA.2009.216
62. Kaveh A, Ghazaan MI (2017) Enhanced whale optimization algorithm for sizing optimization of skeletal structures. Mech Based Des Struct Mach 45(3):345–362. https://doi.org/10.1080/15397734.2016.1213639
63. Hansen P, Mladenovi N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130(3):449–467. https://doi.org/10.1016/S0377-2217(00)00100-4
64. Liang J, Qu B, Suganthan P, G A, (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, pp 3–18
65. Eiben A, Smit S (2011) Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol Comput 1(1):19–31. https://doi.org/10.1016/j.swevo.2011.02.001
66. Carlier J, Pinson E (1989) An algorithm for solving the job-shop problem. Manage Sci 35(2):164–176. https://doi.org/10.1287/mnsc.35.2.164
67. Reeves CR, Yamada T (1998) Genetic algorithms, path relinking, and the flowshop sequencing problem. Evol Comput 6(1):45–60. https://doi.org/10.1162/evco.1998.6.1.45
68. Heller J (1960) Some numerical experiments for an m Œ j flow shop and its decision-theoretical aspects. Oper Res 8(2):178–184. https://doi.org/10.1287/opre.8.2.178
69. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64(2):278–285. https://doi.org/10.1016/0377-2217(93)90182-M

70. Vallada E, Ruiz R, Framinan JM (2015) New hard benchmark for flowshop scheduling problems minimising makespan. Eur J Oper Res 240(3):666–677. https://doi.org/10.1016/j.ejor.2014.07.033
71. Zhao F, Liu Y, Zhang Y, Ma W, Zhang C (2017) A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. Eng Appl Artif Intell 65:178–199. https://doi.org/10.1016/j.engappai.2017.07.023
72. Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18. https://doi.org/10.1016/j.swevo.2011.02.002
73. Luo Q, Zhou Y, Xie J, Ma M, Li L (2014) Discrete bat algorithm for optimal problem of permutation flow shop scheduling. Sci World J
74. Lin Q, Gao L, Li X, Zhang C (2015) A hybrid backtracking search algorithm for permutation flow-shop scheduling problem. Comput Ind Eng 85:437–446. https://doi.org/10.1016/j.cie.2015.04.009
75. Hamdi Imen (2015) Upper and lower bounds for the permutation flowshop scheduling problem with minimal time lags. Optim Lett 9(3):465–482. https://doi.org/10.1007/s11590-014-0761-7
76. Davendra D, Bialic-Davendra M (2013) Scheduling flow shops with blocking using a discrete self-organising migrating algorithm. Int J Prod Res 51(8):2200–2218. https://doi.org/10.1080/00207543.2012.711968
77. Ribas I, Companys R, Tort-Martorell X (2011) An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 39(3):293–301. https://doi.org/10.1016/j.omega.2010.07.007
78. Wang L, Pan QK, Tasgetiren MF (2011) A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. Comput Ind Eng 61(1):76–83. https://doi.org/10.1016/j.cie.2011.02.013
79. Nearchou AC (2004) A novel metaheuristic approach for the flow shop scheduling problem. Eng Appl Artif Intell 17(3):289–300. https://doi.org/10.1016/j.engappai.2004.02.008
80. Zhang C, Ning J, Ouyang D (2010) A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. Comput Ind Eng 58(1):1–11. https://doi.org/10.1016/j.cie.2009.01.016
81. Wang H, Wang W, Sun H, Cui Z, Rahnamayan S, Zeng S (2017) A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems. Soft Comput 21(15):4297–4307. https://doi.org/10.1007/s00500-016-2062-9
82. Deb S, Tian Z, Fong S, Tang R, Wong R, Dey N (2018) Solving permutation flow-shop scheduling problem by rhinoceros search algorithm. Soft Comput 22(18):6025–6034. https://doi.org/10.1007/s00500-018-3075-3

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.