Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.064886

ARTICLE

A Shuffled Frog-Leaping Algorithm with Competition for Parallel Batch Processing Machines Scheduling in Fabric Dyeing Process

Mingbo Li and Deming Lei*

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

*Corresponding Author: Deming Lei. Email: demingleil1@163.com

Received: 26 February 2025; Accepted: 22 April 2025; Published: 30 May 2025

ABSTRACT: As a complicated optimization problem, parallel batch processing machines scheduling problem (PBPMSP) exists in many real-life manufacturing industries such as textiles and semiconductors. Machine eligibility means that at least one machine is not eligible for at least one job. PBPMSP and scheduling problems with machine eligibility are frequently considered; however, PBPMSP with machine eligibility is seldom explored. This study investigates PBPMSP with machine eligibility in fabric dyeing and presents a novel shuffled frog-leaping algorithm with competition (CSFLA) to minimize makespan. In CSFLA, the initial population is produced in a heuristic and random way, and the competitive search of memeplexes comprises two phases. Competition between any two memeplexes is done in the first phase, then iteration times are adjusted based on competition, and search strategies are adjusted adaptively based on the evolution quality of memeplexes in the second phase. An adaptive population shuffling is given. Computational experiments are conducted on 100 instances. The computational results showed that the new strategies of CSFLA are effective and that CSFLA has promising advantages in solving the considered PBPMSP.

KEYWORDS: Batch processing machines; shuffled frog-leaping algorithm; competition; parallel machines scheduling

1 Introduction

Batch processing is a typical production mode and has been widely applied in the manufacturing processes of textiles, chemicals, minerals, pharmaceuticals, semiconductors, and others. Batch processing machines (BPM) can simultaneously process multiple jobs in a batch. Parallel batch and serial batch are often considered, the processing time of the former being the maximum processing time of all jobs in a batch and the processing time of the latter being the sum of the processing time of all jobs in a batch. BPM scheduling problems have attracted much attention [1–5].

Parallel batch processing machines scheduling problem (PBPMSP) is an extended version of the parallel machines scheduling problem [6] and has attracted much attention in the past decades. Koh et al. [7] designed a genetic algorithm (GA) for the problem in a multi-layer ceramic capacitor production line. Su and Wang [8] proposed weighted nested partitions based on differential evolution (DE) for the problem in semiconductor production lines. Zhou et al. [9] developed a multi-objective DE algorithm to solve the problem considering electricity cost. Zhang et al. [10] solved the problem in cloud manufacturing with an efficient algorithm and improved particle swarm optimization (PSO). Gahm et al. [11] handled parallel serial-batch processing machine scheduling by using some heuristics. Zhang et al. [12] presented a mixed integer linear programming (MILP) model and an improved biased random key GA for the problem with two-dimensional bin packing constraints. Ou et al. [13] considered an efficient polynomial time approximation

scheme with near-linear-time complexity. Uzunoglu et al. [14] provided a new multi-start construction heuristic with controlled batch urgencies and a local search mechanism with advanced termination criteria for solution improvement. Kucukkoc et al. [15] dealt with the problem of batch delivery by using a discrete DE algorithm hybridized with GA.

There are some works on uniform PBPMSP. Jula and Leachman [16] gave three algorithms based on linear programming, integer programming, and heuristic for solving the problem in semiconductor manufacturing. Li et al. [17] proposed several heuristics, including an algorithm batching rule and a heuristic assignment rule, to solve the problem of dynamic job arrivals. Some meta-heuristics are applied, which are the Pareto-based ant colony system (Xu et al. [18]), non-dominated sorting genetic algorithm-II (NSGA-II), and multi-objective imperialist competitive algorithm [19], DE [20], ant colony optimization [21,22], artificial immune system [23] and evolutionary algorithm [24].

Non-identical PBPMSP is also solved by PSO [25], GA [26], and fruit fly optimization algorithm [27]. Many results on unrelated PBPMSP were also obtained. Shahidi-Zadeh et al. [28] presented a multi-objective harmony search algorithm. Lu et al. [29] developed a hybrid ABC with tabu search for the problem with maintenance and deteriorating jobs. Zhou et al. [30] solved the problem with different capacities and arbitrary job sizes by a random key GA. Sadati et al. [31] proposed fuzzy multi-objective discrete teaching learning-based optimization and fuzzy NSGA-II. Zarook et al. [32] constructed a MILP model and provided six heuristics and a random key GA. Fallahi et al. [33] studied the problem in production systems under carbon reduction policies, presented a MILP model, NSGA-II and multi-objective gray wolf optimizer. Kong et al. [34] applied a shuffled frog-leaping algorithm (SFLA) with variable neighborhood search for the problem with nonlinear processing time. Xiao et al. [35] proposed a tabu-based adaptive large neighborhood search algorithm, which updates the best-known solutions of 55 instances.

PBPMSP is widely found in various manufacturing processes such as casting [36], food and chemical production [37], semiconductors [38], metal packaging [39], and others. In recent years, PBPMSP in the fabric dyeing process has also received some attention. Zhang et al. [40] proposed a multi-objective artificial bee colony algorithm with an elite retention strategy to simultaneously minimize total tardiness and total completion time. Huynh and Chien [41] studied a hybrid multi-subpopulation GA with heuristic rules. Li et al. [42] proposed a two-objective evolutionary algorithm with the constructive generation of an initial solution and cluster-based environmental selection strategy. Demir [43] developed a lexicographic multi-objective GA to simultaneously minimize total tardiness, total washing times, and total machine fixed costs. Srinath et al. [44] proposed two meta-heuristics to solve multi-objective problems considering SDST and preferences.

Machine eligibility is often investigated in parallel machine scheduling problems [45–47], and there are limited works on PBPMSP with machine eligibility. Li et al. [48] presented a hybrid DE algorithm with chaos theory and two local search algorithms for the problem considering different colors, sequence-dependent setup time (SDST), and machine eligibility. Wang et al. [49] solved a fuzzy energy-efficient problem with machine eligibility and SDST by using a dynamic teaching-learning-based optimization. Lei and Dai [50] dealt with the problem of machine eligibility in the fabric dyeing process by using an adaptive SFLA.

Accordingly, PBPMSP has been studied fully in the last three years, and most of the existing works are related to the usage of heuristics and meta-heuristics; however, PBPMSP with machine eligibility is seldom considered [48–50]. Machine eligibility often exists in real-life manufacturing processes such as fabric dyeing and PBPMSP, with machine eligibility often exhibiting different features compared to the classical PBPMSP. For example, a machine set is used when a machine assignment is done for a job. The inclusion of machine eligibility can make PBPMSP closely resemble the actual situations of manufacturing processes. As a result,

the corresponding optimization results hold high application value, so it is necessary to focus on PBPMSP with machine eligibility in fabric dyeing shops.

Compared to PSO, DE, and GA, SFLA has fast convergence speed and effective algorithm structure containing local search and global information exchanges; in addition, as an effective method for BPM scheduling [34,50], distributed scheduling [51,52], flexible job shop scheduling [53], and flow shop scheduling [54], SFLA has been successfully applied to solve PBPMSP [34,50] and the corresponding results show that has advantages on solving PBPMSP. On the other hand, the integration of new optimization mechanisms and SFLA is an effective way to intensify the search ability of SFLA. Q-learning and cooperation are proven to be useful mechanisms and can improve search advantages of SFLA [52,53]; however, competition among memeplexes is seldom considered. After the competition is added to SFLA, parameters or search strategies can be adjusted in the search process, the search ability can be intensified effectively, and optimization results on PBPMSP can be significantly improved. The advantages of SFLA on PBPMSP and the great impact of competition demonstrate that the shuffled frog-leaping algorithm with competition (CSFLA) is particularly suited for PBPMSP, so it is necessary to focus on CSFLA.

This study considers PBPMSP with machine eligibility in the fabric dyeing process, and a new CSFLA is presented to minimize makespan. The competitive search of memeplexes is composed of two phases to produce high-quality solutions, and the initial population is produced in a heuristic and random way. Competition between any two memeplexes is executed in the first phase, then iteration times are adjusted based on competition, and search strategies are adjusted adaptively based on the evolution quality of memeplexes. An adaptive population shuffling is given. Several computational experiments are conducted on 100 instances. The computational results demonstrate that the new strategies of CSFLA are effective and that CSFLA has promising advantages in solving the PBPMSP considered.

The remaining parts of the paper are organized as follows. The problem description is given in Section 2. CSFLA for PBPMSP is shown in Section 3. Computational results and analyses are provided in Section 4. The conclusions are drawn, and the future research topics are reported in the final section.

2 Problem Description

PBPMSP in fabric dyeing process is composed of n jobs $J_1, J_2, \ldots, J_n$ and m BPM $M_1, M_2, \ldots, M_m$.

Each M_k has a capacity Q_k. There are F families for n jobs in terms of the dyeing color. w_i, d_i, Θ_i and f_i are weight, due date, the set of machines and job family for J_i. All jobs with the same family have the same processing time, p_g indicates processing time of each job in family g. Machine eligibility is considered, which means that at least one machine is not eligible for at least one job, that is, at least one job J_i has $\Theta_i \subset \{M_1, M_2, \ldots, M_m\}$.

All jobs in a batch are processed simultaneously on BPM. Families are incompatible, and jobs within different families cannot be processed in the same batch. Each batch is formed with jobs in the same family. When jobs of family g are applied to form a batch allocated on M_k, job J_i with $M_k \in \Theta_i$ can be chosen and added into batch B_h if the following condition is met.

$$w_i + \sum_{j \in B_h} w_j \leq Q_k$$

The above equation is used described capacity constraint, that is, the sum of weight of job J_i and all jobs in batch B_h cannot exceed Q_k.

All jobs in the same batch are processed with the same beginning time and completion time. When BPM is required to switch from one batch with family g to other batch with family h, setup time is incurred because dyeing machines need cleaning before a different color is handled. st_{gh} indicates setup time between families,

which means that after a batch of family g is processed, the cleaning time is required to process a batch of family h, if two adjacent batches belong to the same family, that is, $g = h$, BPM does not need cleaning.

There are some constraints on jobs and machines.

Each BPM can only process one batch at a time.

No job can be processed in different batches on more than one BPM.

Operations cannot be interrupted.

All BPMs and jobs are always available.

The goal of the problem is to minimize makespan when all constraints are satisfied and three subproblems are solved.

$$C_{\max} = \max\{C_i \,|\, i = 1, 2, \ldots, n\}$$

where C_i is completion time of J_i and $C_{\max}$ is maximum value of $C_1, C_2, \cdots, C_n$.

The above formula is applied to compute $C_{\max}$.

Table 1 lists an illustrative example with 10 jobs, 3 job families and 3 BPMs, $Q_1 = 40$, $Q_2 = 70$, $Q_3 = 90$, $p_1 = 5$, $p_2 = 10$, $p_3 = 15$. The setup time matrix is shown below.

$$\left(st_{ge}\right)_{3\times3} = \begin{bmatrix} 0 & 1 & 3 \\ 4 & 0 & 2 \\ 5 & 1 & 0 \end{bmatrix}$$

In the above equation, $st_{12} = 1$, this setup time exists for the batch of family 1 and batch of family 2, after batch of family 1 is processed, the setup time on batch of family 2 is st_{12}.

Table 1: Related data of example

J_i	w_i	f_i	Θ_i	J_i	w_i	f_i	Θ_i
J_1	50	2	$\{M_3\}$	J_6	20	3	$\{M_1\}$
J_2	10	1	$\{M_3\}$	J_7	40	2	$\{M_1, M_3\}$
J_3	20	2	$\{M_2, M_3\}$	J_8	30	3	$\{M_3\}$
J_4	60	1	$\{M_1, M_2\}$	J_9	10	2	$\{M_2, M_3\}$
J_5	70	3	$\{M_2\}$	J_{10}	80	2	$\{M_1, M_3\}$

3 CSFLA for PBPMSP in the Fabric Dyeing Process

There are s memeplexes $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_s$ in SFLA and the search within each memeplex is implemented independently in most of the previous SFLAs [35,48]. In this study, competition among memeplexes is added, and CSFLA is presented to PBPMSP in the fabric dyeing process.

3.1 Initialization

Zhang et al. [40] presented a two-string representation. For PBPMSP with n jobs, m BPM and F families, its solution is represented as a scheduling string $[\pi_1, \pi_2, \ldots, \pi_n]$ and a machine assignment string $[\theta_1, \theta_2, \ldots, \theta_n]$ for batches, where $\pi_i \in \{1, 2, \ldots, n\}$, $\theta_i \in \{1, 2, \ldots, m\}$.

With respect to $[\theta_1, \theta_2, \ldots, \theta_n]$, if $\eta\,(<n)$ batches are formed, then only $\theta_1, \theta_2, \ldots, \theta_\eta$ are used, other elements are not considered; however, the number of the formed batches is not fixed, so a string with the length of n is used and optimized.

The decoding procedure of [40] is directly adopted. For the example in Table 1, a possible solution is [10, 1, 8, 4, 9, 6, 2, 5, 3, 7] and [2, 1, 3, 3, 2, 3, 3, 1, 1, 2]. After two strings are decoded, a schedule in Fig. 1 is obtained. $B_1 = \{J_1, J_9\}, B_2 = \{J_8\}, B_3 = \{J_{10}\}, B_4 = \{J_2, J_4\}, B_5 = \{J_5\}, B_6 = \{J_6\}, B_7 = \{J_3, J_7\}$. The processing sequence of batches on each BPM is shown in Fig. 1.

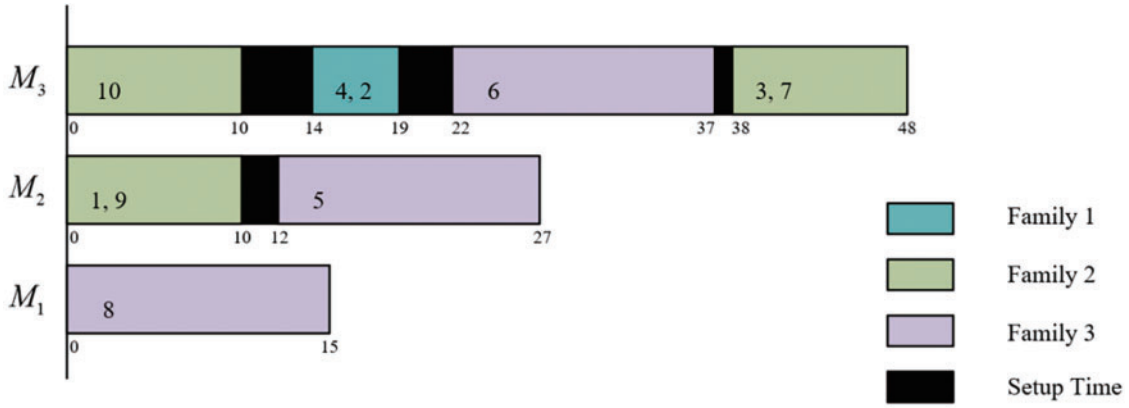

Figure 1: A schedule of the example

Initial population P is composed of $x_1, x_2, \cdots, x_N$. For each $x_i, i = 1, 2, \cdots, n$, if random number $rand < \varepsilon$, then x_i is produced randomly; otherwise, x_i is obtained by using a heuristic, where ε is probability, $rand$ follows uniform distribution on $[0, 1]$.

The heuristic is shown below.

(1) Produce scheduling string by sorting all jobs in the ascending order of w_i.
(2) Calculate the average weight $\overline{w}$ of all jobs, let $t = 1$.
(3) Repeat the following steps until $t > n/2$: stochastically generate an integer $1 \le r \le m$, if $Q_r > \overline{w}$, then let $\theta_t = r$; otherwise, randomly chose one from $m, m-2, m-1$ with the same probability, and let θ_t be the chosen one, $t = t + 1$.
(4) Repeat the following steps until $t > n$: randomly select an integer from $[1, m]$ and let θ_t be the chosen one, $t = t + 1$.

As stated above, jobs with smaller weight are given higher priority to be assigned into the first half of scheduling string, machines with $Q_r > \overline{w}$ are given bigger probability to be allocated in the first half of machine assignment string, so jobs with smaller weight can be formed into a batch and processed on machine with bigger capacity, as a result, the number of the formed batches can be diminished, so maximum completion time can be improved.

Population division is an important step of SFLA. In this study, population $\overline{P}$ is divided into $\overline{s}$ memeplexes in the following way: sort all solutions in $\overline{P}$ in the descending order of $C_{\max}^{x_i}$, suppose that $C_{\max}^{x_1} \le C_{\max}^{x_2} \le \cdots \le C_{\max}^{x_N}$, then allocate each $x_i, i = 1, 2, \cdots .n$ into memeplex $i\,(\mathrm{mod}s) + 1$, finally, s memeplexes $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_s$ are formed, where $i\,(\mathrm{mod}s)$ is the remainder of i/s, $C_{\max}^{x_i}$ is makespan of x_i.

Initially, $\overline{P} = P, \overline{N} = N, \overline{s} = s$.

3.2 Search Operators

Three global search operators GS_1, GS_2, GS_3 are used. GS_1 is described below. For solutions x, y, randomly select $k_1, k_2 \in [1, n]$, $k_1 < k_2$, machines between positions k_1, k_2 on machine assignment string of y substitute for those at the same position on machine assignment string of x. GS_2 is shown as follows. For solutions x, y, order crossover [51] is performed on scheduling string of x, y. GS_3 is depicted below. For solutions x, y, GS_1, GS_2 are executed sequentially.

For solution x with η batches, six neighborhoods' structures are used. Neighborhood structure $\mathcal{N}_1$ is described below: randomly select a π_g and insert it into a newly decided position. $\mathcal{N}_2$ is insertion operator on machine assignment string as done in $\mathcal{N}_1$ $\mathcal{N}_3, \mathcal{N}_4$ are swap operator on scheduling string and machine assignment string, respectively. $\mathcal{N}_5$ is inversion operator for those genes of scheduling string between two randomly decided $k_1, k_2, k_1 < k_2$. $\mathcal{N}_6$ is described below. Decide a machine M_k with the biggest completion time, for each $\theta_l, l = 1, 2, \cdots, \eta$, if $\theta_l = k$, then a randomly chosen integer from $[1, m]$ substitutes for θ_l.

Three search strategies SO_1, SO_2, SO_3 are produced by using global search operators and neighborhood search operators. SO_1 is described below. For solutions x, y, perform GS_1 between x, y and obtain a new solution z_1, if z_1 is better than x, then $x = z_1$, else if z_1 is better than y, then $y = z_1$, else produce $z_2 \in \mathcal{N}_1(z_1)$, then compare z_2 with x, y and replace x or y with the above procedure on z_1, x, y, if z_2 cannot substitutes for y, then generate $z_3 \in \mathcal{N}_1(z_2)$, compare z_2 with x, y and replace x or y with the above procedure on z_1, x, y. Where $\mathcal{N}_g(x)$ indicates the set of neighborhood solutions of x produced by $\mathcal{N}_g$.

When $GS_2, \mathcal{N}_3, \mathcal{N}_4$ are replaced with $GS_1, \mathcal{N}_1, \mathcal{N}_2$ in the above procedure, SO_2 is obtained. SO_3 is also similar with SO_1; however, in SO_3, $GS_3, \mathcal{N}_5, \mathcal{N}_6$ are used sequentially like $GS_1, \mathcal{N}_1, \mathcal{N}_2$ of SO_1.

The initial set S of search strategies consists of SO_1, SO_2, SO_3. The alternate set AS is empty.

3.3 Competitive Search of Memeplexes

In existing SFLAs [34,53], search within memeplex is often done independently; in CSFLA, the competitive search of memeplexes is composed of two phases. In the first phase, any two memeplexes compete with each other. The second phase is done based on the data from the first phase and the adaptive adjustment of search strategies.

Algorithm 1 describes competitive search of memeplexes, where $x_{nearb,i}$ is the best solution of $\mathcal{M}_i \setminus \{x_{b,i}\}$, $x_{b,i}$ is the best solution of $\mathcal{M}_i$, Me_i indicates solution quality of $\mathcal{M}_i$.

$$Me_i = \sum_{x \in \mathcal{M}_i} |\{y \in P | C^x_{\max} < C^y_{\max}\}|$$

In the above equation, for each $x \in \mathcal{M}_i$, the number of solutions of P with smaller $C_{\max}$ than x is computed, then all numbers of all solutions of $\mathcal{M}_i$ are added together and Me_i is obtained.

Algorithm 1: Competitive search of memeplexes

1: for $i = 1$ to s
2: let $R_i = r_i = 0$
3: end for
4: $\Omega_1 = \Omega_2 = \Omega_3$
5: for $i = 1$ to s
6: construct a set $B = \{1 \le j \le s | j \ne i\}$
7: end for

(Continued)

Algorithm 1 (continued)

8: for each $j \in B$
9: decide the best solution $x_{b,i}\left(x_{b,j}\right)$ of $\mathcal{M}_i\left(\mathcal{M}_j\right)$ and $x_{nearb,i}\left(x_{nearb,j}\right)$, execute SO_1, SO_2, SO_3 on $x_{b,i}, x_{nearb,i}$ sequentially, perform SO_1, SO_2, SO_3 on $x_{b,j}, x_{nearb,j}$ sequentially, calculate cnt_i, cnt_j and update Ω_l for SO_l.
10: end for
11: Sort SO_1, SO_2, SO_3 in the descending order of Ω_l, suppose that $\Omega_1 \le \Omega_2 \le \Omega_3$.
12: Sort all memeplexes in the descending order of cnt_i, suppose that $cnt_1 \le cnt_2 \le \cdots \le cnt_s$, calculate solution quality Me_i for each $\mathcal{M}_i$, then $\Upsilon_1 = SO_1$ and compute
$It_1 = 2\left(\mu - 3\left(s-1\right)\right) \times Me_1 / \left(Me_1 + Me_s\right)$ $\Upsilon_s = SO_3$ and compute
$It_s = 2\left(\mu - 3\left(s-1\right)\right) \times Me_s / \left(Me_1 + Me_s\right)$; for other $\mathcal{M}_i, i \ne 1, s$, $It_s = \mu - 3\left(s-1\right)$ $\Upsilon_i = SO_2$
13: let $t = 1$
14: While $t \le It_1$
15: for each $\mathcal{M}_i$, if $t \le It_i$, then decide $x_{b,i}, x_{nearb,i}$ and perform Υ_i on $x_{b,i}, x_{nearb,i}$, when is applied, if a new solution is obtained, $r_i = r_i + 1$ if $x_{b,i}$ is updated with the new solution, $R_i = R_i + 1$
16: for each memeplex $\mathcal{M}_i$, compute $ev_i = r_i / R_i$, decide $ev_{\min}, ev_{\max}, \overline{ev}$, suppose that
$ev_i = ev_{\max}, ev_j = ev_{\min}$
17: if $ev_{\max} - ev_{\min} \le \alpha \times \overline{ev}$, then
18: $\Upsilon_1 = SO_1, \Upsilon_s = SO_3, \Upsilon_i = SO_2, i \ne 1, s$
19: end if
20: if $\alpha \times \overline{ev} < ev_{\max} - ev_{\min} \le \overline{ev}$, then
21: $S = S \cup AS$
22: if Υ_i and Υ_j are different
23: swap them
24: else
25: $\Upsilon_i = SO_3, \Upsilon_j = SO_1$
26: end if
27: end if
28: if $\overline{ev} < ev_{\max} - ev_{\min}$, then
29: let $S = \{SO_1, SO_3\}, AS = \{SO_2\}, \Upsilon_j = SO_1, \Upsilon_l = SO_3, l \ne j$,
30: end if
31: $t = t + 1$
32: end while
33: if AS is not empty
34: $S = S \cup AS$ and let AS be empty
35: end if

In line 9, For $\mathcal{M}_i, \mathcal{M}_j$, let $\gamma_i = \gamma_j = 0$, when SO_l is executed on $x_{b,i}\left(x_{b,j}\right)$ and $x_{b,i}\left(x_{b,j}\right)$ is updated with new solution produced by SO_l, $\gamma_i = \gamma_i + 1\left(\gamma_j = \gamma_j + 1\right)$, after $SO_1 - SO_3$ are executed, if $\gamma_i < (>) \gamma_j$, then $cnt_i = cnt_i + 1, cnt_j = cnt_j - 1\left(cnt_j = cnt_j + 1, cnt_i = cnt_i - 1\right)$.

In line 9, with respect to Ω_l, when SO_l acts on $x_{b,i}$ and $x_{b,i}$ is updated with new solution, then $\Omega_l = \Omega_l + 1$.

In line 12, $\mathcal{M}_1$ is given more iteration times by using some iteration times of $\mathcal{M}_s$ because $\mathcal{M}_1$ wins in the first phase, which is shown in line 9, the search strategy with biggest Ω_1 is used for $\mathcal{M}_1$.

In lines 17–27, when conditions in lines 17 or 20 are met, evolution difference among memeplexes occurs and is avoided by adjusting search strategies for memeplexes.

In the above procedure, competition can lead to more iteration times and SO_1 with the biggest Ω_1 for $\mathcal{M}_1$, the extra iteration times comes from memeplex $\mathcal{M}_s$, as a result, $\mathcal{M}_1$ is used fully and the usage of $\mathcal{M}_s$ is limited, however, in the second phase beginning with line 13, the adjustment on search operator is done to diminish evolution difference among memeplexes, and global search can be improved.

3.4 Algorithm Description

The detailed steps of CSFLA are described as follows.

(1) Produce initial population by using heuristic and random way, for each memeplex $\mathcal{M}_i$, $cnt_i = 0$.
(2) Execute population division.
(3) Perform a competitive search of memeplexes.
(4) Execute adaptive population shuffling.
(5) If the termination condition is not met, then go to step (2); otherwise, stop the search.

The flowchart of CSFLA is shown in Fig. 2.

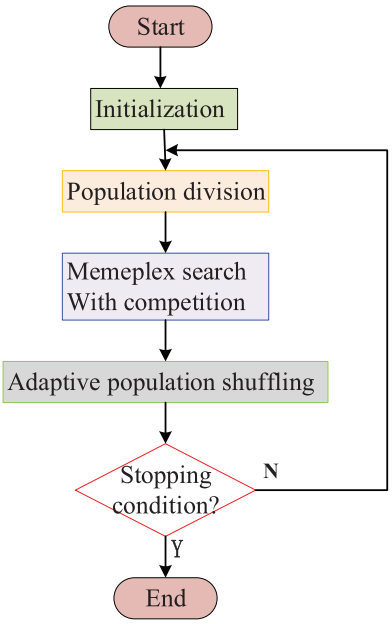

Figure 2: Flowchart of CSFLA

As the main step of SFLA, population shuffling is utilized to form a new population by using all memeplexes, and it is done in each generation. Adaptive population shuffling is proposed and described below. Compute Me_i for each memeplex $\mathcal{M}_i$, $i = 1, 2, \cdots, s$, suppose that memeplex $\mathcal{M}_1$ has the highest Me_i, then let $cnt_i = 0$ for each memeplex $\mathcal{M}_i$, $i \neq 1$, construct a population $\overline{P}$ with all memeplexes except $\mathcal{M}_1$.

In the above process, a memeplex is decided with Me_i and excluded from population $\overline{P}$ and not all memeplexes are utilized to form a new population.

Unlike the existing SFLA [51–54], CSFLA has the following features: (1) Each memeplex $\mathcal{M}_i$ is assigned ev_i, cnt_i, and each strategy SO_l is given Ω_l, these data are utilized to adjust iteration times and search strategy

in an adaptive way. (2) Search procedure of memeplexes is composed of two phases, and the results of the first phase affect the second phase. Competition is implemented in the first phases, and then excessive evolution differences among memeplexes are avoided. (3) An adaptive population shuffling is performed by excluding memeplexes with the highest Me_i.

Competition, Ω_l on SO_l, ev_i, cnt_i for $\mathcal{M}_i$ are added to improve performance of CSFLA, so the implementation of CSFLA using programming language will become more difficult; however, the increasing of implementation difficulty is limited because the newly added things of CSFLA are easily achieved.

4 Computational Experiments

Extensive experiments are conducted to test the performance of CSFLA for the considered PBPMSP. Experiments are implemented by using Microsoft Visual C++ 2019 and run on 8.0 G RAM 2.4 GHz CPU PC.

4.1 Instances and Comparative Algorithms

One hundred instances are generated based on the data refined from the dyeing factory in China. The related data are shown below. $n \in \{100, 200, 300, 400, 500\}$, $m \in \{5, 7, 9, 11, 13\}$, $F \in \{6, 9, 12, 15\}$, $p_g \in [15, 45]$, $st_{gh} \in [4, 9]$, $w_i \in [15, 75]$, $Q_k = 50 + 10k$. Each instance is indicated as $n \times F \times m$. Table 2 describes information on instances.

Table 2: Information on instances

Notation	No.	Notation	No.	Notation	No.
$n \times 6 \times 5$	1, 21, 41, 61, 81	$n \times 9 \times 9$	8, 28, 48, 68, 88	$n \times 12 \times 13$	15, 35, 55, 75, 95
$n \times 6 \times 7$	2, 22, 42, 62, 82	$n \times 9 \times 11$	9, 29, 49, 69, 89	$n \times 15 \times 5$	16, 36, 56, 76, 96
$n \times 6 \times 9$	3, 23, 43, 63, 83	$n \times 9 \times 13$	10, 30, 50, 70, 90	$n \times 15 \times 7$	17, 37, 57, 77, 97
$n \times 6 \times 11$	4, 24, 44, 64, 84	$n \times 12 \times 5$	11, 31, 51, 71, 91	$n \times 15 \times 9$	18, 38, 58, 78, 98
$n \times 6 \times 13$	5, 25, 45, 65, 85	$n \times 12 \times 7$	12, 32, 52, 72, 92	$n \times 15 \times 11$	19, 39, 59, 79, 99
$n \times 9 \times 5$	6, 26, 46, 66, 86	$n \times 12 \times 9$	13, 33, 53, 73, 93	$n \times 15 \times 13$	20, 40, 60, 80, 100
$n \times 9 \times 7$	7, 27, 47, 67, 87	$n \times 12 \times 11$	14, 34, 54, 74, 94		

CSFLA is compared to three existing methods, which are the random key genetic algorithm (RKGA, [30]), RKGA [32], and orthogonal biased random-key genetic algorithm (OBRKGA, [12]) to show the search advantages of CSFLA. RKGA (Zhou et al. [30]) is called as RKGA1, and RKGA (Zarook et al. [32]) as RKGA2 for convenience. These GAs are utilized to solve PBPMSP with the minimization of makespan and can be directly applied to solve the PBPMSP; in addition, three GAs have promising advantages on solving the problems in references [12,30,32], so they are chosen.

SFLA was constructed to show the impact of new strategies of CSFLA on its performance. The steps of SFLA are identical with the general SFLA, when memeplex search is done, one of SO_1, SO_2, SO_3 is randomly selected as search strategy between two solutions, for example, x_w, x_b. Where x_w, x_b are the worst solutions and the best ones in a memeplex.

4.2 Parameter Settings

CSFLA has the following parameters: N, s, μ, α and stopping condition. In this study, CPU time is used as a stopping condition. The study found through experiments that CSFLA, SFLA, and three comparative

algorithms can converge fully on all instances when $0.05 \times n$ s CPU time reaches, so this CPU time is used as a stopping condition.

Taguchi method [55] is applied to obtain the settings of parameters. Instance 50 is used. Table 3 describes the levels of each parameter. The orthogonal array $L_{16}\left(4^4\right)$ is executed. 16 parameter combinations are tested. CSFLA with each parameter combination runs 10 times for the chosen instance.

Table 3: Parameter level

Parameter	Factor level			
	1	2	3	4
N	30	60	90	120
s	3	5	6	10
μ	20	30	40	50
α	0.2	0.4	0.6	0.8

Fig. 3 shows the results of MIN and S/N ratio, which is defined as $-10\log_{10}\left(MIN^2\right)$. CSFLA runs 10 times for the selected instance. In the run, an elite solution is obtained. MIN is the best one of 10 elite solutions obtained in 10 runs. Fig. 3 indicates that CSFLA with following combination $N = 90$, $s = 10$, $\mu = 50$, $\alpha = 0.2$ can obtain better results than CSFLA with other combinations, so the above parameter settings are used.

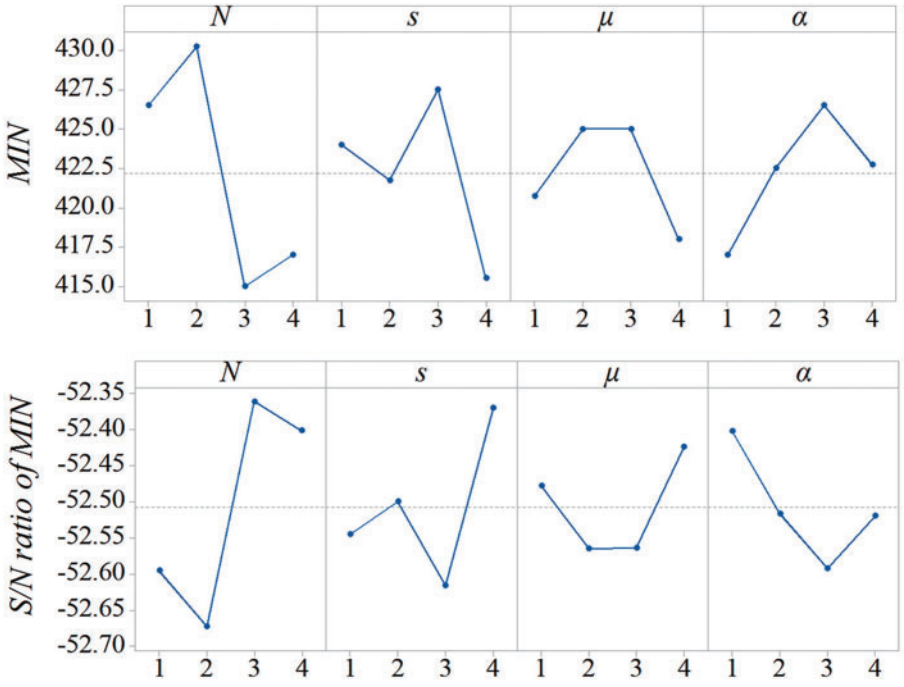

Figure 3: Main effect plot for MIN and S/N ratio

SFLA has $N = 90$, $s = 10$, $\mu = 50$ and the above stopping condition.

With respect to RKGA1, RKGA2, and OBRKGA, their parameter settings, except the stopping condition, are directly used in this study. The experimental results show that these settings of each comparative algorithm are still effective, so they are kept. Three GAs are given the same stopping condition as CSFLA.

4.3 Results and Analyses

CSFLA, SFLA, and three RKGA are compared, each of which randomly runs 10 times on each instance. In a run, an elite solution is obtained, MAX indicates the worst one of 10 elite solutions, and AVG denotes the average value of 10 elite solutions. Tables 4–6 show the computational results of five algorithms. CS, SF, RK1, RK2, OB indicate CSFLA, SFLA, RKGA1, RKGA2, OBRKGA. Fig. 4 describes the convergence curves of all algorithms, and Fig. 5 provides a box plot.

Table 4: Computational results of five algorithms on metric MIN

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
1	535	532	532	646	563	51	1552	2440	2435	1572	1645
2	906	909	1132	1714	997	52	2320	3329	2411	3228	2347
3	1126	1389	1401	1474	1561	53	2952	3926	3791	3042	3018
4	922	1142	1047	1102	1074	54	3291	4247	4201	3581	3559
5	783	791	813	774	869	55	3636	4613	4526	4078	4192
6	566	568	570	624	658	56	2345	2381	2408	2462	2467
7	852	867	878	1017	1035	57	3105	3181	3179	3389	3347
8	1040	1062	1134	1241	1278	58	3717	3731	3760	4091	3758
9	676	691	684	688	734	59	4214	4244	4276	4768	4477
10	545	587	600	591	614	60	4627	4625	4442	5249	4535
11	478	490	487	541	585	61	2191	2639	1714	2568	1709
12	803	819	760	772	835	62	2734	3681	2879	3694	3481
13	505	531	544	518	518	63	3493	4633	3623	4595	4459
14	604	643	618	630	605	64	4150	5213	4232	5006	5050
15	716	719	729	742	717	65	4612	5766	4731	5820	5664
16	516	543	529	615	559	66	2803	2990	2811	2810	2829
17	790	855	833	834	840	67	3923	4174	4222	4085	4016
18	902	941	927	940	940	68	4815	5138	4851	4953	4846
19	841	871	903	861	860	69	5356	5745	5520	5611	5563
20	959	1117	963	1092	1092	70	5798	6067	5927	6080	6160
21	1068	1083	1068	1124	1118	71	3252	3361	3206	3197	2966
22	1618	1714	1665	1774	1860	72	4198	4580	4273	4326	4365
23	2001	2172	2097	2765	2709	73	5036	5468	5238	5093	5102
24	2287	2463	2471	2453	2566	74	5543	6028	5742	5739	5770
25	2612	2650	2625	2637	2749	75	6015	6491	6080	6240	6326
26	909	937	950	929	981	76	3216	3292	3003	3152	3037
27	1462	1479	1565	1594	1892	77	4333	4458	4421	4428	4368
28	1832	1849	1849	2190	2178	78	4954	5226	5038	5278	5044
29	2179	2093	2150	2429	2478	79	5532	5771	5659	5807	6050
30	2080	2186	2243	2139	2221	80	6087	6314	6263	6246	6497

(Continued)

Table 4 (continued)

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
31	1153	1284	1158	1167	1271	81	3779	4023	3877	3890	3908
32	1744	1926	1755	1864	2248	82	5305	5133	5130	5223	5212
33	2094	2359	2176	2903	2347	83	6036	6203	6156	6137	6321
34	2404	2681	2456	2418	2684	84	6808	7036	6949	6873	6895
35	1818	1863	1881	1880	1875	85	7344	7319	7454	7531	7324
36	923	963	953	951	985	86	3703	3955	3766	3903	4087
37	1531	1593	1560	1793	1614	87	4985	5361	5264	5122	5022
38	1943	1943	1968	2478	2159	88	5904	6400	6071	5945	6513
39	2284	2313	2319	2471	2756	89	6543	6959	6751	7085	6602
40	1902	1911	1980	1905	2010	90	7139	7635	7500	7018	7133
41	1485	1535	1536	1504	1490	91	3427	3746	3796	3594	3571
42	2197	2427	2343	2456	2391	92	4734	5116	4970	4844	5074
43	2767	2867	2899	2934	3009	93	5537	6154	5876	6115	5720
44	3250	3385	3354	3445	3307	94	6572	7001	6687	6598	6583
45	3637	3647	3557	3681	3586	95	6921	7401	7131	7263	7129
46	1351	2121	1968	1376	1699	96	3694	4076	3903	3892	3806
47	2124	2154	2790	2902	2547	97	5063	5573	5223	5288	5168
48	2807	2808	3378	3579	3523	98	6102	6550	6252	6425	6269
49	3071	3913	3760	3368	3326	99	7023	7399	7233	7031	7029
50	3671	4335	4402	4733	3988	100	7571	7939	7765	7530	7536

Table 5: Computational results of five algorithms on metric MAX

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
1	553	566	575	677	654	51	1636	2486	2659	2530	2413
2	964	925	1863	2162	1694	52	2459	3362	3409	3477	2828
3	1355	1466	1745	1599	1738	53	3089	4130	3931	3153	3578
4	1302	1394	1428	1437	1680	54	3321	4265	4404	3851	3660
5	831	873	902	926	1021	55	3710	4664	4630	4256	4287
6	585	588	592	648	705	56	2405	2405	2514	2578	2715
7	925	930	898	1600	1573	57	3200	3247	3285	3566	3425
8	1121	1156	1474	1377	1363	58	3817	3851	3875	4230	3917
9	721	756	732	779	822	59	4299	4307	4350	4911	4618
10	629	694	686	663	742	60	4703	4801	4739	5608	4916
11	511	590	591	561	641	61	2546	2694	2852	2624	2667
12	837	849	846	855	964	62	3721	3865	3853	3778	3764
13	571	628	579	624	566	63	4508	4691	4604	4649	4645
14	673	691	731	716	691	64	5096	5287	5232	5305	5612
15	760	821	775	816	833	65	5510	5818	5776	5877	6563
16	555	568	598	657	659	66	3060	3067	2915	2844	2989
17	851	902	871	903	957	67	4039	4295	4341	4221	4077

(Continued)

Table 5 (continued)

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
18	1023	1088	963	1015	1057	68	5169	5161	4923	5010	5100
19	1030	1083	1136	1181	1124	69	5548	5912	5807	5781	5747
20	1201	1283	1335	1219	1268	70	5946	6174	6027	6244	6272
21	1082	1119	1107	1152	1215	71	3326	3423	3432	3299	3234
22	1754	1757	1703	1955	2437	72	4300	4681	4417	4459	4625
23	2114	2211	2162	3050	3046	73	5215	5534	5351	5226	5594
24	2440	2482	2594	2711	2785	74	5709	6052	5861	5796	6089
25	2716	2732	2837	2761	2830	75	6162	6608	6242	6549	6529
26	988	1565	987	964	1498	76	3336	3408	3090	3278	3132
27	1511	2120	1646	1630	1956	77	4398	4636	4482	4611	4432
28	1875	2545	1876	2522	2767	78	5137	5341	5475	5414	5163
29	2328	2484	2377	3052	3206	79	5737	6024	5819	5913	6104
30	2249	2354	2573	2622	2343	80	6228	6405	6349	6556	6623
31	1199	1674	1272	1206	1378	81	4012	4057	4044	4089	4093
32	1756	2278	1871	1949	2545	82	5371	5297	5342	5324	5324
33	2251	2766	2280	3005	2479	83	6161	6263	6269	6262	6391
34	2497	3134	2663	2521	3062	84	6973	7295	7010	6999	7240
35	2000	2085	2075	2061	2127	85	7469	7837	7557	7616	7776
36	966	1069	1057	1067	1138	86	3865	4214	4016	4046	4128
37	1585	1646	1652	1839	1958	87	5147	5451	5483	5517	5380
38	1989	2003	2079	2681	2414	88	6089	6535	6246	6126	6547
39	2412	2440	2538	3068	3169	89	6725	7275	6867	7369	7238
40	2142	2214	2260	2306	2115	90	7414	7758	7658	7775	7506
41	1538	1562	1639	1563	1789	91	3547	3761	3816	3719	3613
42	2363	2464	2385	2498	2673	92	4907	5253	5101	4983	5138
43	2804	2911	2926	3056	3116	93	5808	6255	5966	6398	6243
44	3283	3453	3468	3568	3483	94	6602	7072	6772	6876	6759
45	3654	3841	3721	4010	3769	95	7583	7519	7273	7329	7427
46	2003	2244	2254	2238	2113	96	3752	4125	4050	4117	4004
47	2837	2951	2963	2978	2876	97	5104	5698	5327	5417	5224
48	3425	3534	3620	3774	3961	98	6349	6714	6548	6631	6370
49	3268	4137	3936	4333	3712	99	7079	7530	7349	7092	7340
50	4353	4459	4792	5023	4489	100	7677	8015	7925	7857	7759

Table 6: Computational results of five algorithms on metric AVG

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
1	545.4	550.7	562.5	660.2	580.3	51	1562.5	2457.6	2505.3	1724.7	1828.1
2	938.3	916.4	1838.0	1984.1	1650.1	52	2442.5	3351.8	3398.9	3378.0	2570.5
3	1221.3	1423.5	1643.9	1576.1	1702.9	53	3010.6	4023.5	3816.7	3064.8	3471.0
4	1041.4	1353.3	1209.8	1264.2	1574.3	54	3300.1	4255.6	4288.3	3820.7	3598.6

(Continued)

Table 6 (continued)

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
5	808.1	809.5	872.4	811.4	952.4	55	3666.0	4643.0	4564.2	4174.6	4246.5
6	576.7	578.0	580.6	636.0	687.1	56	2358.0	2393.3	2470.7	2479.3	2505.2
7	865.5	901.1	888.5	1390.2	1129.0	57	3143.6	3198.8	3258.6	3523.3	3397.9
8	1104.4	1075.1	1269.2	1356.9	1352.0	58	3762.0	3761.2	3811.5	4215.7	3904.9
9	711.8	729.9	719.4	728.3	808.5	59	4276.2	4281.2	4338.9	4890.0	4533.8
10	592.8	610.5	671.3	602.9	723.5	60	4658.3	4776.6	4643.9	5295.5	4754.9
11	497.8	580.3	508.3	551.9	617.8	61	2215.1	2681.8	2371.9	2601.6	1886.3
12	820.4	830.1	829.1	840.0	948.4	62	3015.8	3831.5	3277.1	3759.4	3590.0
13	558.6	586.6	569.0	542.0	535.9	63	4478.5	4648.3	4531.5	4629.8	4480.1
14	620.2	656.3	638.7	654.2	624.6	64	4825.4	5250.2	4402.6	5254.4	5090.2
15	732.2	781.3	743.3	791.3	822.6	65	4928.9	5798.3	4846.4	5859.2	6171.8
16	526.1	555.4	560.7	644.3	577.3	66	2942.6	3031.8	2894.3	2820.8	2841.8
17	819.6	877.4	857.0	882.3	893.0	67	3962.9	4185.1	4242.8	4168.4	4038.0
18	934.5	1049.2	945.4	982.9	958.1	68	5072.3	5151.7	4883.3	4993.9	4867.2
19	864.1	1039.0	1126.8	893.5	908.4	69	5445.0	5798.3	5552.4	5688.9	5681.9
20	1155.8	1212.2	1111.4	1146.1	1104.8	70	5846.3	6113.6	5972.8	6234.3	6262.5
21	1075.0	1099.2	1089.3	1142.8	1183.3	71	3260.6	3390.7	3354.0	3240.3	3014.8
22	1729.0	1725.2	1679.7	1816.4	2073.9	72	4254.8	4612.4	4344.2	4344.7	4407.1
23	2026.0	2185.0	2118.0	3020.3	2745.1	73	5061.2	5518.1	5258.2	5166.1	5436.4
24	2388.8	2477.1	2551.0	2629.3	2753.2	74	5557.3	6041.8	5787.3	5755.3	5897.1
25	2630.1	2716.2	2722.7	2660.1	2820.8	75	6059.2	6543.9	6225.9	6353.8	6454.1
26	955.8	982.5	967.4	945.9	1377.8	76	3328.5	3328.4	3044.3	3204.0	3054.6
27	1488.1	1985.2	1616.2	1609.0	1914.8	77	4380.4	4523.4	4444.7	4555.0	4415.1
28	1862.1	2485.4	1865.1	2253.8	2489.7	78	5076.5	5298.9	5176.2	5362.9	5118.5
29	2261.1	2224.2	2346.9	2792.5	3074.3	79	5640.6	5814.3	5672.8	5889.0	6077.4
30	2166.8	2298.0	2399.0	2261.8	2258.9	80	6130.4	6343.2	6307.2	6307.6	6573.1
31	1189.8	1491.9	1204.1	1195.1	1283.2	81	3802.9	4043.7	4025.1	3941.0	4069.9
32	1749.1	2202.8	1820.2	1884.2	2336.8	82	5341.6	5174.6	5257.9	5296.8	5247.2
33	2147.2	2731.8	2234.8	2932.0	2426.6	83	6115.2	6226.4	6256.0	6161.6	6337.4
34	2432.3	2875.0	2584.0	2498.7	3047.0	84	6884.9	7256.7	6968.3	6920.0	7064.0
35	1847.0	1928.2	1978.7	2047.3	1886.9	85	7387.1	7503.0	7499.9	7606.8	7614.3
36	938.1	1015.7	1046.9	1007.8	1076.4	86	3739.6	4082.5	3919.7	4027.7	4112.9
37	1540.9	1610.0	1594.3	1827.1	1644.7	87	5004.3	5389.0	5374.6	5334.8	5298.6
38	1955.6	1966.7	2009.2	2585.8	2255.5	88	5990.4	6477.5	6098.2	6031.7	6523.9
39	2307.1	2422.1	2439.3	2805.1	2976.8	89	6627.9	7176.2	6763.2	7294.5	6612.4
40	2025.4	2084.1	2057.7	2290.7	2093.8	90	7405.1	7710.3	7538.1	7593.3	7447.6
41	1527.3	1547.3	1574.2	1540.9	1501.6	91	3518.9	3752.2	3806.9	3658.6	3587.6
42	2285.9	2440.7	2356.2	2469.1	2528.7	92	4839.1	5202.7	5083.3	4940.2	5111.1
43	2782.0	2897.2	2909.5	2977.8	3057.7	93	5741.2	6174.9	5921.8	6236.3	5988.5
44	3269.7	3415.8	3389.0	3529.4	3327.4	94	6595.5	7048.8	6736.1	6664.0	6692.3
45	3647.3	3665.3	3586.1	3892.3	3725.0	95	7226.7	7509.5	7150.6	7286.0	7204.1

(Continued)

Table 6 (continued)

No.	CS	SF	RK1	RK2	OB	No.	CS	SF	RK1	RK2	OB
46	1924.4	2188.4	2181.8	1953.8	2027.0	96	3709.1	4102.7	4032.2	4096.2	3844.0
47	2183.5	2347.7	2877.2	2944.5	2818.8	97	5071.0	5686.9	5258.7	5361.2	5203.7
48	3339.8	3326.1	3594.9	3644.1	3863.3	98	6257.6	6702.1	6530.1	6474.0	6360.3
49	3259.4	4061.0	3837.8	3833.8	3490.6	99	7049.9	7449.1	7285.1	7066.8	7240.2
50	3881.7	4391.1	4766.8	4747.8	4117.9	100	7649.6	7961.2	7885.4	7822.2	7546.4

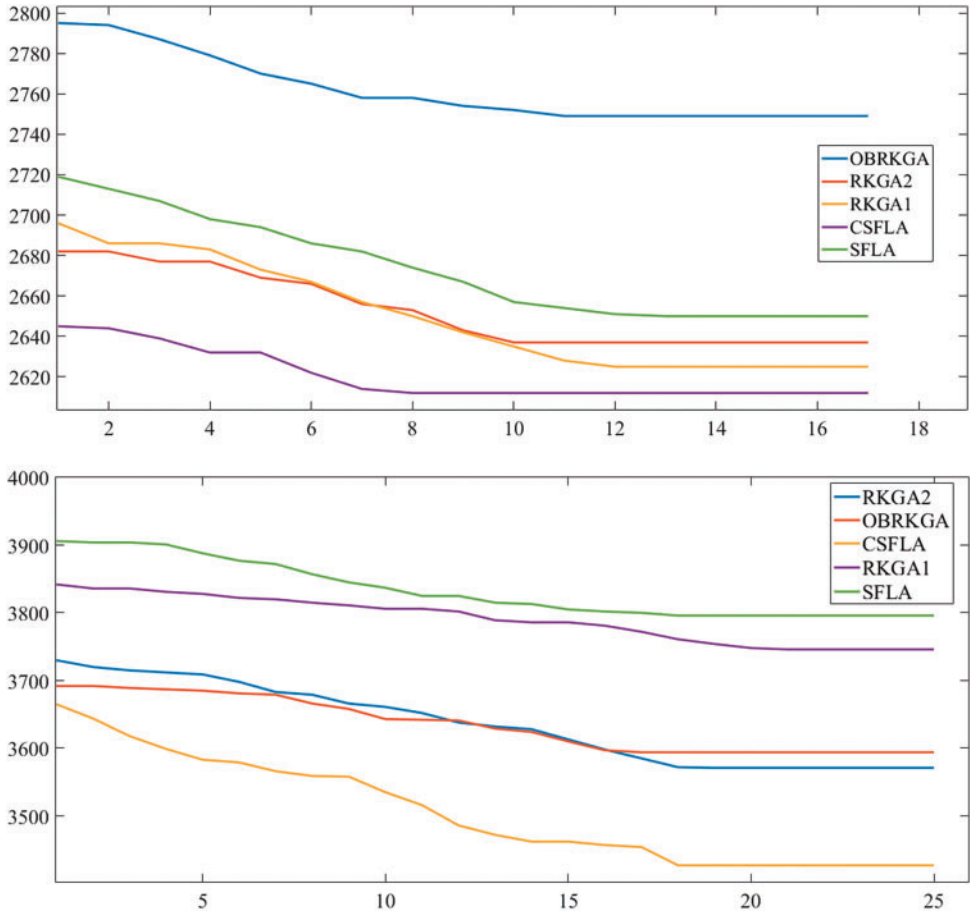

Figure 4: Convergence curves of five algorithms

Table 4 indicates that CSFLA obtains a smaller MIN than SFLA on 94 instances, and the MIN of CSFLA is bigger than that of SFLA just on 5 instances; in addition, the MIN of CSFLA is less than by at least 100 on 57 instances. CSFLA significantly outperforms SFLA on convergence. The convergence curves in Fig. 4 and $RPD_{\min}$ in Fig. 5 also reveal that CSFLA converges better than SFLA.

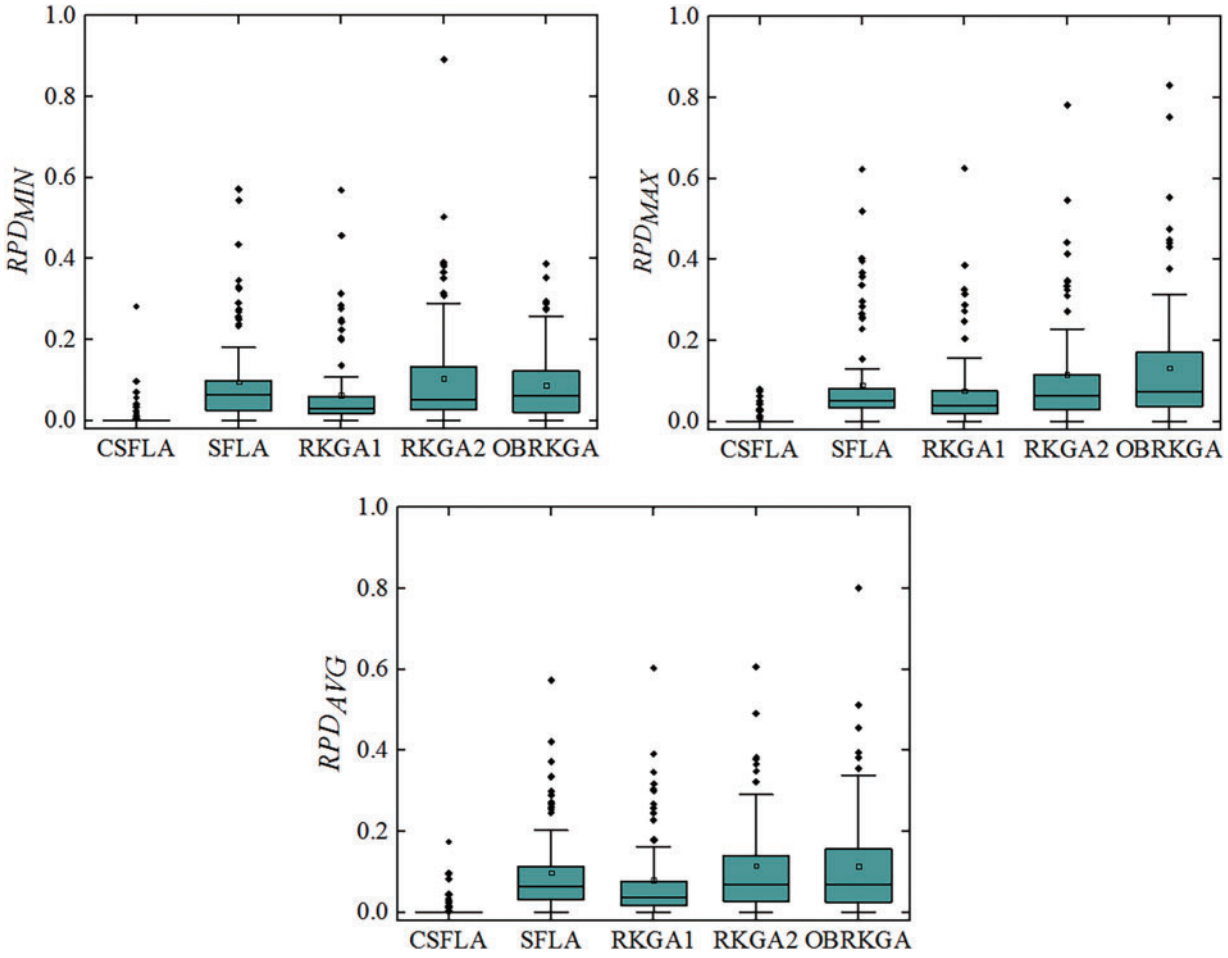

Figure 5: Box plot of all algorithms

Tables 5 and 6 indicate that CSFLA performs better than SFLA on MAX and AVG. SFLA just produces a smaller MAX than CSFLA in 4 instances, and there are differences between the MAX of CSFLA and SFLA. CSFLA obtains smaller AVG on 93 instances. The performance advantages of CSFLA on MAX and AVG also can be observed in Fig. 5. The above analyses show new strategies.

When CSFLA is compared to RKGA1, it can be found that CSFLA produces better results than RKGA1. RKGA1 obtains a smaller MIN than CSFLA in six instances; however, the MIN of RKGA1 is worse than that of CSFLA in 94 of 100 instances. Obviously, CSFLA converges better than RKGA1.

Competition and excessive evolution differences really have a positive impact on the performance of CSFLA. Thus, these new strategies are effective.

Convergence differences between CSFLA and RKGA1 also can be observed in Figs. 4 and 5. Tables 5 and 6 indicate that CSFLA performs better than RKGA1 on MAX, AVG, MAX of CSFLA is less than that of RKGA1 by at least 100 on 54 instances, and CSFLA also obtains smaller AVG than RKGA1 on 93 instances.

Tables 4–6 exhibit that CSFLA has a smaller MIN than RKGA2 and OBRKGA on 89 instances, the MAX of CSFLA is smaller than that of RKGA2 and OBRKGA on 91 instances, and CSFLA has better average results than RKGA2 and OBRKGA more than 85 instances. CSFLA really performs better than RKGA2 and

OBRKGA, and this conclusion can also be concluded from Figs. 4 and 5. Based on the above analyses, it can be concluded that CSFLA outperforms three GAs on three metrics.

The above analyses show that the superiority of CSFLA lies in the performance between CSFLA and its comparative algorithm. The advantage of CSFLA results from its adaptive adjustment of search strategies and iteration times by using competition and avoiding excessive evolution differences. With the inclusion of these new strategies, global search ability can be intensified, and a good solution structure can be kept in the memeplex. As a result, search efficiency is significantly improved. Thus, CSFLA is a competitive method for solving PBPMSP.

5 Conclusion

This study considers PBPMSP with machine eligibility in the fabric dyeing process and presents a new algorithm called CSFLA to minimize makespan. Population initialization is implemented in a heuristic and random way. A competitive search of memeplexes is composed of two phases. Competition between any two memeplexes is done in the first phase, then iteration times are adjusted based on competition, and search strategies are adjusted adaptively based on the evolution quality of memeplexes. An adaptive population shuffling is also given. Computational experiments are conducted, and experimental results validate the effectiveness of new strategies and the promising advantages of CSFLA in solving the considered PBPMSP in the fabric dyeing process.

BPM extensively exists in many real-life manufacturing industries, and the BPM scheduling problem has higher complexity than the classical scheduling problem. Shortly, it will focus on PBPMSP, a hybrid flow shop scheduling problem with BPM. In recent years, learning, cooperation, feedback, and competition have been included in meta-heuristics. Meta-heuristics with new optimization mechanisms for scheduling problems are also future research topics.

Acknowledgement: Thanks to anonymous reviewers and the editors for providing valuable suggestions for the paper.

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

Author Contributions: The authors confirm their contribution to the paper as follows: Conceptualization: Deming Lei; methodology, formal analysis, original draft preparation: Mingbo Li; writing—review and editing: Deming Lei. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data generated in this paper are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

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

References

1. Azizoglu M, Webster S. Scheduling a batch processing machine with non-identical job sizes. Int J Prod Res. 2000;38(10):2173–84. doi:10.1080/00207540050028034.
2. Shi ZS, Huang ZW, Shi LY. Customer order scheduling on batch processing machines with incompatible job families. Int J Prod Res. 2018;56(1–2):795–808. doi:10.1080/00207543.2017.1401247.
3. Trindade RS, de Araujo OCB, Fampa MHC, Muller FM. Modelling and symmetry breaking in scheduling problems on batch processing machines. Int J Prod Res. 2018;56(22):7031–48. doi:10.1080/00207543.2018.1424371.

4. Zheng SX, Xie NM, Wu Q, Liu CJ. Novel mathematical formulations for parallel-batching processing machine scheduling problems. Comput Oper Res. 2025;173(20):106859. doi:10.1016/j.cor.2024.106859.
5. Wang J, Lei DM, Tang HT. A multi-objective dynamical artificial bee colony for energy-efficient fuzzy hybrid flow shop cheduling with batch processing machines. Expert Syst Appl. 2025;259(2):125244. doi:10.1016/j.eswa.2024.125244.
6. Esfandiari S, Mashreghi H, Emami S. Coordination of order acceptance, scheduling, and pricing decisions in unrelated parallel machine scheduling. Int J Ind Eng Prod Res. 2019;30(2):195–205.
7. Koh SG, Koo PH, Ha JW, Lee WS. Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families. Int J Prod Res. 2004;42(19):4091–107. doi:10.1080/00207540410001704041.
8. Su GJ, Wang XH. Weighted nested partitions based on differential evolution (WNPDE) algorithm-based scheduling of parallel batching processing machines (BPM) with incompatible families and dynamic lot arrival. Int J Comput Integ Manuf. 2011;24(6):552–60. doi:10.1080/0951192X.2011.562545.
9. Zhou SC, Li XL, Du N, Pang Y, Chen HP. A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost. Comput Oper Res. 2018;96(1):55–68. doi:10.1016/j.cor.2018.04.009.
10. Zhang H, Li K, Chu CB, Jia ZH. Parallel batch processing machines scheduling in cloud manufacturing for minimizing total service completion time. Comput Oper Res. 2022;146(2):105899. doi:10.1016/j.cor.2022.105899.
11. Gahm C, Wahl S, Tuma A. Scheduling parallel serial-batch processing machines with incompatible job families, sequence-dependent setup times and arbitrary sizes. IntJ Prod Res. 2022;60(17):5131–54. doi:10.1080/00207543.2021.1951446.
12. Zhang XP, Shan MJ, Zeng J. Parallel batch processing machine scheduling under two-dimensional bin-packing constraints. IEEE Trans Reliab. 2023;72(3):1265–75. doi:10.1109/TR.2022.3201333.
13. Ou JW, Lu LF, Zhon XL. Parallel-batch scheduling with rejection: structural properties and approximation algorithms. Euro J Oper Res. 2023;310(3):1017–32. doi:10.1016/j.ejor.2023.04.019.
14. Uzunoglu A, Gahm C, Wahl S, Tuma A. Learning-augmented heuristics for scheduling parallel serial-batch processing machines. Comput Oper Res. 2023;151(3):106122. doi:10.1016/j.cor.2022.106122.
15. Kucukkoc I, Keskin GA, Karaoglan AD, Karadag S. A hybrid discrete differential evolution—genetic algorithm approach with a new batch formation mechanism for parallel batch scheduling considering batch delivery. Int J Prod Res. 2024;62(1–2):460–82. doi:10.1080/00207543.2023.2233626.
16. Jula P, Leachman RC. Coordinated multistage scheduling of parallel batch-processing machines under multiresource constraints. Oper Res. 2010;58(4):933–47. doi:10.1287/opre.1090.0788.
17. Li XL, Chen HP, Du B, Tan Q. Heuristics to schedule uniform parallel batch processing machines with dynamic job arrivals. Int J Comput Integ Manuf. 2013;26(5):474–86. doi:10.1080/0951192X.2012.731612.
18. Xu R, Chen HP, Li XP. A bi-objective scheduling problem on batch machines via a Pareto-based ant colony system. Int J Prod Econo. 2013;145(1):371–86. doi:10.1016/j.ijpe.2013.04.053.
19. Abedi M, Seidgar H, Fazlollahtabar H, Bijani R. Bi-objective optimisation for scheduling the identical parallel batch-processing machines with arbitrary job sizes, unequal job release times and capacity limits. Int J Prod Res. 2015;53(6):1680–711. doi:10.1080/00207543.2014.952795.
20. Zhou SC, Liu M, Chen HP, Li XP. An effective discrete differential evolution algorithm for scheduling uniform parallel batch processing machines with non-identical capacities and arbitrary job sizes. Int J Prod Econo. 2016;179(1):1–11. doi:10.1016/j.ijpe.2016.05.014.
21. Jia ZH, Yan JH, Leung JYT, Li K, Chen HP. Ant colony optimization algorithm for scheduling jobs withfuzzy processing times on parallel batch machines with different capacities. Appl Soft Comput. 2019;75(9–10):548–61. doi:10.1016/j.asoc.2018.11.027.
22. Jia ZH, Huo SY, Li K, Chen HP. Integrated scheduling on parallel batch processing machines with non-identical capacities. Eng Optim. 2020;52(4):715–30. doi:10.1080/0305215X.2019.1613388.
23. Li CH, Wang F, Gupta JND, Chung T. Scheduling identical parallel batch processing machines involving incompatible families with different job sizes and capacity constraints. Comput Ind Eng. 2022;169(5):108115. doi:10.1016/j.cie.2022.108115.

24. Li K, Zhang H, Chu CB, Jia ZH, Chen JF. Abi-objective evolutionary algorithm scheduled on uniform parallel batch processing machines. Exp Syst Appl. 2022;204(6):117487. doi:10.1016/j.eswa.2022.117487.
25. Liu M, Chu F, He JK, Yang DP, Chu CB. Coke production scheduling problem: a parallel machine scheduling with batch preprocessings and location-dependent processing times. Comput Oper Res. 2019;104(7):37–48. doi:10.1016/j.cor.2018.12.002.
26. Hulett M, Damodaran P, Amouie M. Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. Comput Ind Eng. 2017;113(8):425–36. doi:10.1016/j.cie.2017.09.037.
27. Wang R, Jia ZH, Li K. Scheduling parallel-batching processing machines problem with learning and deterioration effect in fuzzy environment. J Intel Fuzzy Syst. 2021;40(6):12111–24. doi:10.3233/JIFS-210196.
28. Shahidi-Zadeh B, Tavakkoli-Moghaddam R, Taheri-Moghadam A, Rastgar I. Solving a bi-objective unrelated parallel batch processing machines scheduling problem: a comparison study. Comput Oper Res. 2017;88(6):71–90. doi:10.1016/j.cor.2017.06.019.
29. Lu SJ, Liu XB, Pei J, Thai MT, Pardalos PM. A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. Appl Soft Comput. 2018;66(2):168–82. doi:10.1016/j.asoc.2018.02.018.
30. Zhou SC, Xie JH, Du N, Pang Y. A random-keys genetic algorithm for scheduling unrelated parallel batch processing machines with different capacities and arbitrary job sizes. Appl Math Comput. 2018;334(8):254–68. doi:10.1016/j.amc.2018.04.024.
31. Sadati A, Moghaddam RT, Naderi B, Mohammadi M. Abi-objective model for a scheduling problem of unrelated parallel batch processing machines with fuzzy parameters by two fuzzy multi-objective meta-heuristics. Iran J Fuzzy Syst. 2019;16(4):21–40.
32. Zarook Y, Rezaeian J, Mahdavi I, Yaghini M. Efficient algorithms to minimize makespan of the unrelated parallel batch-processing machines scheduling problem with unequal job ready times. Performa Analy Intel Syst Oper. 2021;55(3):1501–22. doi:10.1051/ro/2021062.
33. Fallahi A, Shahidi-Zadeh B, Niaki STA. Unrelated parallel batch processing machine scheduling for production systems under carbon reduction policies: nSGA-II and MOGWO metaheuristics. Soft Comput. 2023;27(22):17063–91. doi:10.1007/s00500-023-08754-0.
34. Kong M, Wang WZ, Deveci M, Zhang YJ, Wu XZ, Coffman D. A novel carbon reduction engineering method-based deep Q-learning algorithm for energy-efficient scheduling on a single batch-processing machine in semiconductor manufacturing. Int J Prod Res. 2024;62(18):6449–72. doi:10.1080/00207543.2023.2252932.
35. Xiao X, Ji B, Yu SS, Wu GH. A tabu-based adaptive large neighborhood search for scheduling unrelated parallel batch processing machines with non-identical job sizes and dynamic job arrivals. Flex Serv Manuf J. 2024;36(2):409–52. doi:10.1007/s10696-023-09488-9.
36. Zhang W, Tang HT, Wang WY, Zhuang M, Lei D, Wang XV. A multi-objective hybrid algorithm for the casting scheduling problem with unrelated batch processing machine. Compl Syst Model Simul. 2024;4(3):236–57. doi:10.23919/CSMS.2024.0011.
37. Mageed G, Sharareh T. Dynamic shop-floor scheduling using real-time information: a case study from the thermoplastic industry. Comput Oper Res. 2023;152(4):106134. doi:10.1016/j.cor.2022.106134.
38. Hu KX, Che YX, Ng TS, Deng J. Unrelated parallel batch processing machine scheduling with time requirements and two-dimensional packing constraints. Comput Oper Res. 2024;162(3):106474. doi:10.1016/j.cor.2023.106474.
39. Fu LL, Aloulou MA, Triki C. Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows. Int J Prod Res. 2017;55(20):5942–57. doi:10.1080/00207543.2017.1308572.
40. Zhang R, Chang PC, Song SJ, Wu C. A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes. Knowl-Based Syst. 2017;116(7):114–29. doi:10.1016/j.knosys.2016.10.026.
41. Huynh NT, Chien CF. A hybrid multi-subpopulation genetic algorithm for textile batch dyeing scheduling and an empirical study. Comput Ind Eng. 2018;125(2):615–27. doi:10.1016/j.cie.2018.01.005.

42. Li K, Zhang H, Chu CB, Jia ZH, Wang Y. A bi-objective evolutionary algorithm for minimizing maximum lateness and total pollution cost on non-identical parallel batch processing machines. Comput Ind Eng. 2022;172(3):108608. doi:10.1016/j.cie.2022.108608.
43. Demir Y. An efficientlexicographic approach to solve multi-objective multi-port fabric dyeing machine planning problem. Appl Soft Comput. 2023;144(2):110541. doi:10.1016/j.asoc.2023.110541.
44. Srinath N, Yilmazlar IO, Kurz ME, Taaffe K. Hybrid multi-objective evolutionary meta-heuristics for a parallel machine scheduling problem with setup times and preferences. Comput Ind Eng. 2023;185(5–6):109675. doi:10.1016/j.cie.2023.109675.
45. Lee DH, Park IB, Kim K. An incremental learning approach to dynamical parallel machine scheduling with sequence-dependent setups and machine eligibility restrictions. Appl Soft Comput. 2024;164(21):112002. doi:10.1016/j.asoc.2024.112002.
46. Santoro MC, Junqueira L. Unrelated parallel machine scheduling models with machine avaiability and eligibility constraints. Comput Ind Eng. 2023;179(1–2):109219. doi:10.1016/j.cie.2023.109219.
47. Zheng FF, Jin KY, Xu YF, Liu M. Unrelated parallel machine scheduling with processing cost, machine eligibility and order splitting. Comput Ind Eng. 2022;171(3):108483. doi:10.1016/j.cie.2022.108483.
48. Li DB, Wang J, Qiang R, Chiong R. A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility. Int J Prod Res. 2021;59(9):2722–38. doi:10.1080/00207543.2020.1740341.
49. Wang J, Li DB, Tang HT, Li XX, Lei DM. A dynamical teaching-learning-based optimization algorithm for fuzzy energy-efficient parallel batch processing machines scheduling in fabric dyeing process. Appl Soft Comput. 2024;167(9):112413. doi:10.1016/j.asoc.2024.112413.
50. Lei DM, Dai T. An adaptive shuffled frog-leaping algorithm for parallel batch processing machines scheduling with machine eligibility in fabric dyeing process. Int J Prod Res. 2024;62(21):7704–21. doi:10.1080/00207543.2024.2324452.
51. Cai JC, Lei DM. A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time. Comple Intel Syst. 2021;7(5):2235–53. doi:10.1007/s40747-021-00400-2.
52. Cai JC, Lei DM, Wang J, Wang L. A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling. Int J Prod Res. 2022;61(4):1233–51. doi:10.1080/00207543.2022.2031331.
53. Yang YF, Song YH, Guo WF, Lei Q, Sun AH, Fan LH. Guided shuffled frog-leaping algorithm for flexible job shop scheduling with variable sublots and overlapping in operations. Comput Ind Eng. 2023;180(19):109209. doi:10.1016/j.cie.2023.109209.
54. Wu PL, Yang QY, Chen WB, Mao BY, Yu HN. An improved genetic-shuffled frog-leaping algorithm for permutation flowshop scheduling. Complexity. 2020;1(14):3450180. doi:10.1155/2020/3450180.
55. Taguchi G. Introduction to quality engineering: designing quality into products and processes. Tokyo, Japan: Asian Productivity Organization; 1986. 191 p.