Complex & Intelligent Systems (2021) 7:2235–2253
https://doi.org/10.1007/s40747-021-00400-2

ORIGINAL ARTICLE

A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time

Jingcao Cai[1] · Deming Lei[1]

Received: 28 December 2020 / Accepted: 11 May 2021 / Published online: 28 May 2021

Abstract
Distributed hybrid flow shop scheduling problem (DHFSP) has attracted some attention; however, DHFSP with uncertainty and energy-related element is seldom studied. In this paper, distributed energy-efficient hybrid flow shop scheduling problem (DEHFSP) with fuzzy processing time is considered and a cooperated shuffled frog-leaping algorithm (CSFLA) is presented to optimize fuzzy makespan, total agreement index and fuzzy total energy consumption simultaneously. Iterated greedy, variable neighborhood search and global search are designed using problem-related features; memeplex evaluation based on three quality indices is presented, an effective cooperation process between the best memeplex and the worst memeplex is developed according to evaluation results and performed by exchanging search times and search ability, and an adaptive population shuffling is adopted to improve search efficiency. Extensive experiments are conducted and the computational results validate that CSFLA has promising advantages on solving the considered DEHFSP.

Keywords Distributed scheduling · Hybrid flow shop scheduling · Fuzzy scheduling · Shuffled frog-leaping algorithm · Energy consumption

Introduction

Production scheduling is a decision-making process that plays an important role in manufacturing and production systems. In the past decades, scheduling problems have been extensively investigated in single factory by exact method, heuristic and meta-heuristics. In recent years, production has shifted from single factory to multiple factories to quickly respond to market changes and customer demands and distributed scheduling in multiple factories has become the main topic of the current scheduling researches. A number of results have been obtained on distributed scheduling in various production environments [2–4,21,21].

Hybrid flow shop scheduling problems (HFSP) extensively exist in many manufacturing industries such as electronics, textile and semiconductor [41]. In recent years, distributed hybrid flow shop scheduling problem (DHFSP) has been successfully considered and some progresses are made. For DHFSP in homogeneous factories, Ying and Lin [51] studied DHFSP with multiprocessor tasks and presented a mixed integer programming model and a self-tuning IG to minimize makespan. Hao et al. [13] proposed a hybrid brain storm optimization algorithm to optimize makespan. Cai et al. [6] developed a dynamic SFLA with dynamic search process for DHFSP with multiprocessor tasks. Cai et al. [5] addressed bi-objective DHFSP and gave a shuffled frog-leaping algorithm with memeplex quality. Shao et al. [43] proposed a DNEH with smallest-medium rule and a multi-neighborhood IG to minimize makespan. Zheng et al. [54] provided a cooperative coevolution algorithm with problem-specific strategies by combining estimation of distribution algorithm and IG for fuzzy DHFSP. Li et al. [29] designed a discrete artificial bee colony algorithm for DHFSP with sequence dependent setup time. Regarding DHFSP in heterogeneous factories, Lei and Wang [21] presented a shuffled frog-leaping algorithm (SFLA) with memeplex grouping for distributed two-stage HFSP with the minimization of makespan and the number of tardy jobs. Cai et al. [7] presented a collaborative variable search to solve fuzzy distributed scheduling in two-stage hybrid flow shop with sequence-dependent setup time. Li et al. [28] presented an

✉ Deming Lei
deminglei11@163.com

[1] School of Automation, Wuhan University of Technology, Wuhan, China

improved artificial bee colony algorithm for DHFSP. Wang and Wang [45] propose a bi-population cooperative memetic algorithm for solving DHFSP to minimize makespan.

As an important energy-efficient scheduling problem, energy-efficient HFSP plays an essential part in green manufacturing. In recent years, it has attracted much attention. Dai et al. [8] developed a genetic-simulated annealing algorithm to minimize makespan and total energy consumption. Luo et al. [34] presented a novel ant colony optimization for HFSP with the consideration on electricity consumption cost. Tang et al. [44] developed an improved particle swarm optimization for energy-efficient dynamic scheduling. Lin et al. [32] presented teaching-learning-based optimization (TLBO) for the integration of processing parameter optimization and HFSP with the minimization of makespan and carbon footprint. Yan et al. [50] proposed a multi-level optimization method for reducing the total energy consumption and makespan. Lei et al. [17] provided a novel TLBO to minimize total tardiness regarded as key one and total energy consumption. Li et al. [26] presented an energy-aware multi-objective optimization algorithm for HFSP with setup energy consumptions. Zeng et al. [52] applied a hybrid non-dominated sorting genetic algorithm-II for flexible flow shop scheduling with total electricity consumption and material wastage. Meng et al. [35] proposed an improved GA with a new energy-conscious decoding method. Li et al. [27] presented a two-level imperialist competitive algorithm.

In general, DHFSP exists in many real-world manufacturing situations like wafer production in more than one wafer fab and the consideration of DHFSP can result in greater improvements in production performances than that of HFSP in a single factory. It can be found that the previous works have the following three features.

(1) Homogeneous factories are frequently considered and heterogeneous factories are seldom handled. Because the former is an idea case and the latter extensively exits in the real-life production process, it is vital to investigate DHFSP in heterogeneous factories. (2) Energy-efficient HFSP has attracted much attention in single factory; however, the energy-related objective or constraints are hardly considered in DHFSP. Obviously, energy-saving of multiple factories production is more important than that of single factory production, so energy-efficient DHFSP should be studied fully. (3) Distributed scheduling with uncertainty attracted limited attention in hybrid flow shop [7,54]. Uncertainty always exists in the real-life manufacturing process and is an unavoidable property of manufacturing process [1]. The obtained schedule may be invalid if uncertainty is neglected in scheduling problem; moreover, the negligence of uncertainty in multiple factories will produce more serious consequences than that in single factory. Thus, it can be concluded from the above analyses that it is important to study DEHFSP with uncertainty in heterogeneous factories.

In the past decade, fuzzy theory is often adopted in fuzzy scheduling to depict uncertainty and a number of works have been done on fuzzy scheduling problem [1,16,18,25,30,31,33,47] in single factory and fuzzy distributed scheduling in multiple factories [7,54]. Some works on DHFSP with fuzzy processing conditions are also done; however, the related researches are not performed fully.

SFLA is a meta-heuristic by observing, imitating and modelling the behavior of a group of frogs searching for a location with the maximum amount of available food [10]. It has fast convergence speed, effective algorithm structure containing local search and global information exchanges. SFLA has been used to handle parallel machine scheduling, flow shop scheduling and HFSP etc by discretization in single factory [11,15,22,23,37–39,49,53]. There are some applications of SFLA to distributed scheduling [6,21]. Not only that, SFLA performs very well in solving other scheduling problems [46]. The optimization abilities and advantages of SFLA have been validated on scheduling problems.

In the previous works, memeplexes evolve independently after population is divided into s memeplexes and the cooperation between memeplexes is seldom considered based on the quality evaluation of memeplex. For example, the best memeplex often consists of good solutions and the worst memeplex has some worse solutions. The cooperation between these two memeplexes can avoid the waste of computing resources on the worst memeplex and make full use of good search ability of the best memeplex.

In this study, an cooperated shuffled frog-leaping algorithm (CSFLA) is proposed to address fuzzy DEHFSP in heterogeneous factories. The goal is to optimize fuzzy makespan and fuzzy total energy consumption and total agreement index simultaneously. In CSFLA, IG, VNS and global search (GS) are designed based on features of the problem and memeplex evaluation is done in terms of three quality indices: solution quality, evolution quality and contribution degree of archive; to improve search efficiency, an effective cooperation between the best memeplex and the worst memeplex is performed in search process of memeplexes and an adaptive population shuffling is presented. CSFLA is tested on a number of benchmark instances and compared with the methods from literature. The computational results demonstrate that the new strategies of CSFLA are effective and efficient and CSFLA can generate better results than its comparative algorithms.

The remained parts of the paper are organized as follows. Operations on fuzzy scheduling is described in the second section followed by problem description in third section. Introduction to SFLA is shown in fourth section. CSFLA for DEHFSP with fuzzy processing time is given in fifth section. Computational results and analyses are provided in sixth section. The conclusions are drawn and the future research topics are reported in the final section.

Table 1 Notations and descriptions

Notation	Description
i, f, l, k	Indexes
n	The number of jobs
J_i	The i-th job
F	The number of factories
m	The number of stages
w_{fl}	The number of parallel machines at stage l of factory f
M_{flk}	The kth machine at stage l of factory f
$\tilde{p}_{iflk}$	Fuzzy processing time of J_i on M_{flk}, $\tilde{p}_{iflk} = \left(p^1_{iflk}, p^2_{iflk}, p^3_{iflk}\right)$
$\widetilde{d}_i$	Fuzzy due date of J_i, $\widetilde{d}_i = \left(d^1_i, d^2_i\right)$
E_{flk}	Energy consumption per unit time in processing mode
SE_{flk}	Energy consumption per unit idle time
AI_i	The agreement index of J_i
TAI	The total agreement index of all jobs
$\widetilde{C}_i$	The completion time of J_i
$C_{\max}$	The maximum completion time of all jobs
TEC	Fuzzy total energy consumption
Φ_{flk}	The set of all jobs processed on M_{flk}
ID_{flk}	The idle time of M_{flk}
$\widetilde{U}$	TFN $\widetilde{U} = \{U_1, U_2, U_3\}$,where U_1, U_2, U_3 are very large positive numbers
X_{if}	Decision variable, if J_i is allocated in factory f, $X_{if} = 1$; otherwise $X_{if} = 0$
Y_{iflk}	Decision variable, if J_i is allocated in M_{iflk}, $Y_{flk} = 1$; otherwise $Y_{iflk} = 0$
$Z_{ii'fl}$	Decision variable, if J_i is processed before $J_{i'}$ at stage l in factory f, $Z_{ii'fl} = 1$; otherwise $Z_{ii'fl} = 0$
$\widetilde{\mathrm{st}}_{il}$	The start time of process of J_i at stage l
$\widetilde{\mathrm{et}}_{il}$	The end time of process of J_i at stage l
P	Population
N	The number of solutions of P
s	The number of memeplexes
μ	The baseline of search times of memeplex
Evo_i	The evolution quality of $\mathcal{M}_i$
Sol_i	The solution quality of $\mathcal{M}_i$
Con_i	The contribution degree for archive Ω
Me_i	The quality of $\mathcal{M}_i$
$\mathcal{Q}$	The memory solution set
Ω	The non-dominated solution set
η	The search times of memeplexes
R	The maximum search times of VNS1 and VNS2
γ	A real number which equals to 0.5

Operations on fuzzy number

Triangular fuzzy number (TFN) is often used to indicate the processing time in fuzzy scheduling because of its simple operations. Some operations include addition operation, the ranking methods of TFNs and max operation of two TFNs are needed to build a fuzzy schedule.

Addition is used to decide completion time of job. For TFNs $\tilde{s} = (s_1, s_2, s_3)$ and $\tilde{t} = (t_1, t_2, t_3)$, addition operation is performed by

$$\tilde{s} + \tilde{t} = (s_1 + t_1, s_2 + t_2, s_3 + t_3) \tag{1}$$

For $\tilde{s}$, $c_1(\tilde{s}) = (s_1 + 2s_2 + s_3)/4$, $c_2(\tilde{s}) = s_2$, $c_3(\tilde{s}) = s_3 - s_1$.

Ranking of $\tilde{s}$ and $\tilde{t}$ is done according to the following conditions:

If $c_1(\tilde{s}) > c_1(\tilde{t})$, then $\tilde{s} > \tilde{t}$. If $c_1(\tilde{s}) = c_1(\tilde{t})$ and $c_2(\tilde{s}) > c_2(\tilde{t})$, then $\tilde{s} > \tilde{t}$. If $c_i(\tilde{s}) = c_i(\tilde{t}), i = 1, 2$ and $c_3(\tilde{s}) > c_3(\tilde{t})$, then $\tilde{s} > \tilde{t}$.

Ranking is adopted to compare fuzzy objectives of solutions.

Max operation is applied to compute beginning time of job [16] and defined by

$$\text{if } \tilde{s} > \tilde{t}, \text{ then } \tilde{s} \vee \tilde{t} = \tilde{s}; \text{ otherwise, } \tilde{s} \vee \tilde{t} = \tilde{t}, \tag{2}$$

where $\tilde{s} \vee \tilde{t}$ denotes the larger one of $\tilde{s}$ and $\tilde{t}$.

Multiplication operation is applied to compute total energy consumption in this study. For TFN $\tilde{s} = (s_1, s_2, s_3)$ and a constant α, multiplication operation is defined by

$$\alpha \times \tilde{s} = (\alpha \times s_1, \alpha \times s_2, \alpha \times s_3)\,. \tag{3}$$

Problem description

DEHFSP with fuzzy processing time can be described as follows. There are n jobs distributed among F heterogeneous factories located in different sites. Each factory is a hybrid flow shop with m stages and there are w_{fl} unrelated parallel machines at stage l of each factory f. All jobs are available at time zero. Each machine M_{flk} exists two modes: processing mode and stand-by mode.

DEHFSP has some constraints on jobs and machines:

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

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

Operations cannot be interrupted.

All machines are available at all times.

DEHFSP can be categorized into three sub-problems: factory assignment deciding a factory of each job, machine assignment and scheduling. There are strong coupling relationships among these sub-problems [6].

The mathematical model of DEHFSP with fuzzy processing time, which is extended from Wang and Wang [45], is as follows:

$$\text{Maximize } \text{TAI} = \sum_{i=1}^{n} \text{AI}_i, \tag{4}$$

$$\text{Minimize } C_{\max} = \max_{i=1,2,\ldots,n} \widetilde{C}_i, \tag{5}$$

$$\text{Minimize } \text{TEC} = \sum_{f=0}^{F} \sum_{l=0}^{m} \sum_{k=0}^{w_{fl}} \left(\sum_{J_i \in \Phi_{flk}} \tilde{p}_{iflk} \left(E_{flk} - \text{SE}_{flk} \right) + \text{SE}_{flk} \times \max_{J_i \in \Phi_{flk}} \{\widetilde{C}_i\} \right), \tag{6}$$

$$\sum_{f=1}^{F} X_{if} = 1, \forall i, \tag{7}$$

$$\sum_{k=1}^{w_{fl}} Y_{iflk} = X_{if}, \forall i, f, l, \tag{8}$$

$$\widetilde{\text{st}}_{i1} \geq \{0, 0, 0\}, \forall i, \tag{9}$$

$$\widetilde{\text{st}}_{i(l+1)} \geq \widetilde{\text{et}}_{il}, \forall i, l, \tag{10}$$

$$\widetilde{\text{et}}_{il} = \widetilde{\text{st}}_{il} + \sum_{f=1}^{F} \sum_{l=1}^{w_{fl}} \left(\tilde{p}_{iflk} \times X_{if} \times Y_{iflk} \right), \forall i, l, \tag{11}$$

$$Z_{ii'fl} + Z_{i'ifl} \leq 1, \forall f, l, i, i', \tag{12}$$

$$Z_{ii'fl} + Z_{i'ifl} \geq Y_{iflk} + Y_{i'flk} - 1, \forall f, l, i' > i, \tag{13}$$

$$\widetilde{st}_{i'l} \geq \widetilde{et}_{il} - \widetilde{U} \times \left(3 - Y_{iflk} - Y_{i'flk} - Z_{ii'fl}\right), \forall i \neq i', f, l, k \in \{1, 2, \ldots, w_{fl}\}, \tag{14}$$

$$X_{if} \in \{0, 1\}, \forall i, f, \tag{15}$$

$$Y_{iflk} \in \{0, 1\}, \forall i, f, l, k \in \{1, 2, \ldots, w_{fl}\}, \tag{16}$$

$$Z_{ii'fl} \in \{0, 1\}, \forall i, i', f, l, \tag{17}$$

where Eq. (4) is to maximize total agreement index; Eq. (5) is to minimize fuzzy makespan; Eq. (6) is to minimize fuzzy total energy consumption; constraint (7) indicates that each job is processed on only one factory; constraint (8) shows that each job can only be processed in one machine at every stage; constraint (9) demonstrates that each job can be processed after zero time; constraint (10) indicates that the start time of process at stage $l + 1$ is not earlier than the end time of process at stage l; constraint (11) shows that the process cannot be interrupted; constraint (12)–(14) demonstrate that each machine can only process one job at one time; constraint (15)–(17) give the binary decision variables.

Regarding TAI, it is the sum of AI_i of all jobs. Agreement index is defined by Sakawa and Kubota [42] and used to define the degree of tardiness of $\widetilde{C}_i$ to due date $\widetilde{d}_i$. Figure 1 shows fuzzy makespan, fuzzy due date and agreement index. AI_i is defined by

$$\text{AI}_i = \text{area}\left(\widetilde{C}_i \cap \widetilde{d}_i\right) / \text{area}\left(\widetilde{C}_i\right) \tag{18}$$

With respect to TEC, for machine M_{flk}, its completion time is equal to the sum of processing times of all jobs in Φ_{flk} plus the sum ID_{flk} of idle time, that is,

$$\max_{J_i \in \Phi_{flk}} \left\{\widetilde{C}_i\right\} = \sum_{J_i \in \Phi_{flk}} \tilde{p}_{iflk} + \text{ID}_{flk}. \tag{19}$$

EC_{flk} is energy consumption of M_{flk}, Obviously,

$$\text{EC}_{flk} = \sum_{J_i \in \Phi_{flk}} \tilde{p}_{iflk} \times E_{flk} + \text{ID}_{flk} \times \text{SE}_{flk}. \tag{20}$$

TEC can be obtained after the above two equations are combined together and is adopted to avoid computing ID_{flk}

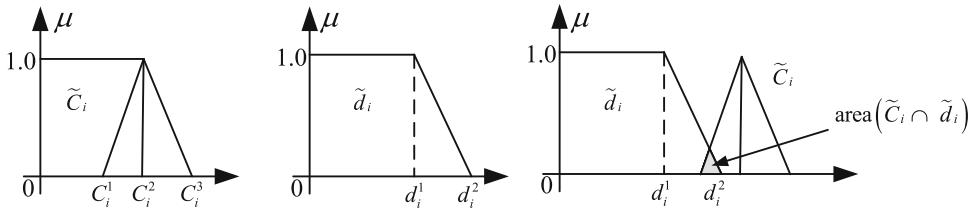


Fig. 1 A schedule of the example

Algorithm 1 SFLA

1: Population initialization
2: **while** stopping condition is not met **do**
3: Population division
4: Memeplex search.
5: Population shuffling.
6: **end while**

because ID_{flk} is related to the subtraction of TFNs and difference between two TFNs is not TFNs frequently and hard to be obtained.

Introduction to SFLA

SFLA was first proposed by Eusuff et al. [10]. SFLA starts with an initial population P, in which each solution is represented as the position of a frog. Algorithm 1 shows the process of SFLA.

Population division is to divide the population into s memeplexes to form multiple sub-populations, which described as follows. All solutions are sorted in the descending order of fitness, then g-th solution is allocated into memeplex $(g-1)(\mathrm{mod}\, s)+1$, where $g(\mathrm{mod}\, s)$ indicates the remainder of g/s .

Memeplex search is the main process to find a better solution. Search within memeplex $\mathcal{M}_t$ is shown below. The following steps are repeated η times: an optimization object x_w is first chosen, then a new solution x' is produced by using x_w and x_b by Eq. 19, if the new one is better than x_w, then replace x_w with x'; otherwise, a solution x' is generated by Eq. 19 after x_g substitutes for x_b, if x' has better fitness than x_w, then x' becomes x_w; otherwise, x_w is replaced with a randomly obtained solution, x_w, x_b and x_g are the worst solution in memeplex, the best solution in memeplex and the best solution of P.

$$x' = x_w + \mathrm{rand} \times (x_b - x_w), \tag{21}$$

where rand is a random number following uniform distribution in [0,1].

Population shuffling is done in the following way. All evolved memeplexes $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_s$ are combined together and a new population is formed.

CSFLA for DEHFSP

In SFLA, search within memeplex is often executed independently and there are no considerations on cooperation between memeplexes. Cooperation may be an effective way to intensify search ability of SFLA. In CSFLA, all memeplexes are evaluated and sorted, and then cooperation between the best memeplex and the worst one is implemented and an adaptive population shuffling is done. We describe these strategies in the following sections.

Initialization and population division

DEHFSP is composed of three sub-problems: factory assignment, machine assignment and scheduling. A two-string representation is used. For DEHFSP with n jobs and F factories, a solution is denoted by a factory assignment string $[\theta_1, \theta_2, \ldots, \theta_n]$ and a scheduling string $[\pi_1, \pi_2, \ldots, \pi_n]$. $\theta_i \in \{1, 2, \ldots, F\}$ indicates the factory allocated for job J_i, $\pi_i \in \{1, 2, \ldots, n\}$. Machine assignment is decided by a rule.

The decoding procedure is described below. All allocated jobs in each factory are decided according to factory assignment string and then permutation of these allocated jobs in each factory is obtained in terms of scheduling string, then for job permutation of each factory f, start with the first job, for each job J_i at stage l, choose a machine M_{flk} with minimum completion time and allocate J_i on M_{flk}.

Table 1 gives an example of DEHFSP. A possible solutions is composed of [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1] and [20, 18, 5, 16, 15, 3, 14, 7, 19, 2, 8, 1, 17, 6, 11, 4, 10, 13, 9, 12]. Figure 1 describes the schedule of the solution.

Initial population P with N solutions is randomly produced.

Population division is implemented as follows. A population $\bar{P}$ is first constructed, then $g = 1$, binary tournament is performed repeatedly until $\bar{P}$ is empty: two solutions $x, y \in \bar{P}$ are randomly chosen, if $x \succ y (y \succ x)$, then add $x(y)$ into memeplex $\mathcal{M}_g$, and delete $x(y)$ from $\bar{P}$; otherwise, randomly choose one of x and y, add the chosen solution into $\mathcal{M}_g$ and remove it from $\bar{P}$. $g = g+1, g = (g-1)(\mathrm{mod}\, \bar{s})+1$.

$x \succ y$ means x dominates y and is defined as follows: $x \succ y$ if $\mathrm{TAI}(x) \geq \mathrm{TAI}(y)$, $C_{\max}(x) \leq C_{\max}(y)$, $\mathrm{TEC}(x) \leq$

Table 2 Fuzzy processing time for 20 jobs on 2 factories

Job	Machine							
	M_{111}	M_{112}	M_{113}	M_{121}	M_{211}	M_{212}	M_{221}	M_{222}
J_1	(74,74,82)	(68,68,78)	(67,75,76)	(67,68,83)	(70,80,82)	(52,61,74)	(60,64,82)	(52,60,62)
J_2	(54,62,64)	(53,62,62)	(53,62,67)	(65,77,77)	(63,67,75)	(72,76,96)	(69,71,78)	(70,74,81)
J_3	(65,75,77)	(64,75,82)	(52,60,64)	(60,63,64)	(60,61,67)	(65,70,73)	(59,66,70)	(62,68,83)
J_4	(68,77,82)	(62,67,70)	(76,77,77)	(59,67,75)	(78,80,100)	(67,75,86)	(60,66,66)	(64,69,81)
J_5	(65,77,85)	(64,71,75)	(64,76,90)	(66,70,82)	(65,76,76)	(65,75,94)	(71,77,94)	(60,67,72)
J_6	(66,70,75)	(56,64,64)	(68,72,77)	(54,63,77)	(52,61,62)	(58,64,78)	(53,60,68)	(58,64,65)
J_7	(68,77,87)	(77,79,80)	(54,61,77)	(64,67,76)	(63,64,73)	(66,75,76)	(52,61,62)	(65,67,79)
J_8	(57,63,69)	(61,65,67)	(62,63,71)	(61,68,79)	(71,77,78)	(71,72,77)	(65,76,85)	(74,79,80)
J_9	(56,64,71)	(68,78,81)	(76,80,86)	(70,72,73)	(67,69,74)	(64,68,70)	(68,75,80)	(67,72,81)
J_{10}	(69,69,79)	(60,61,74)	(65,66,77)	(56,67,71)	(67,68,76)	(68,74,87)	(75,76,80)	(63,73,84)
J_{11}	(55,62,68)	(65,76,80)	(70,73,90)	(59,69,73)	(67,70,88)	(66,75,95)	(59,67,78)	(64,71,83)
J_{12}	(69,70,78)	(60,68,78)	(59,68,72)	(61,61,75)	(68,80,84)	(69,75,79)	(64,71,74)	(62,71,75)
J_{13}	(68,80,101)	(71,80,85)	(64,65,74)	(59,66,68)	(72,75,82)	(67,69,74)	(54,60,60)	(73,77,83)
J_{14}	(66,76,80)	(65,66,77)	(59,67,73)	(63,69,85)	(64,75,92)	(58,68,75)	(60,67,68)	(60,61,64)
J_{15}	(62,71,81)	(65,69,84)	(59,69,70)	(58,62,66)	(69,70,77)	(76,80,83)	(64,67,74)	(69,69,72)
J_{16}	(63,73,82)	(59,68,70)	(60,67,78)	(61,72,87)	(77,79,81)	(70,78,83)	(62,63,65)	(63,72,87)
J_{17}	(71,73,85)	(68,79,88)	(56,60,77)	(65,71,79)	(61,71,71)	(66,72,77)	(59,65,72)	(61,62,74)
J_{18}	(62,66,83)	(55,61,62)	(65,68,85)	(54,60,73)	(56,60,75)	(76,77,80)	(72,79,83)	(60,69,74)
J_{19}	(66,69,83)	(55,63,66)	(65,69,75)	(61,70,72)	(56,62,76)	(62,68,81)	(72,76,77)	(70,75,89)
J_{20}	(61,68,87)	(65,67,75)	(64,72,75)	(75,78,90)	(56,62,71)	(61,67,76)	(64,73,76)	(63,74,82)

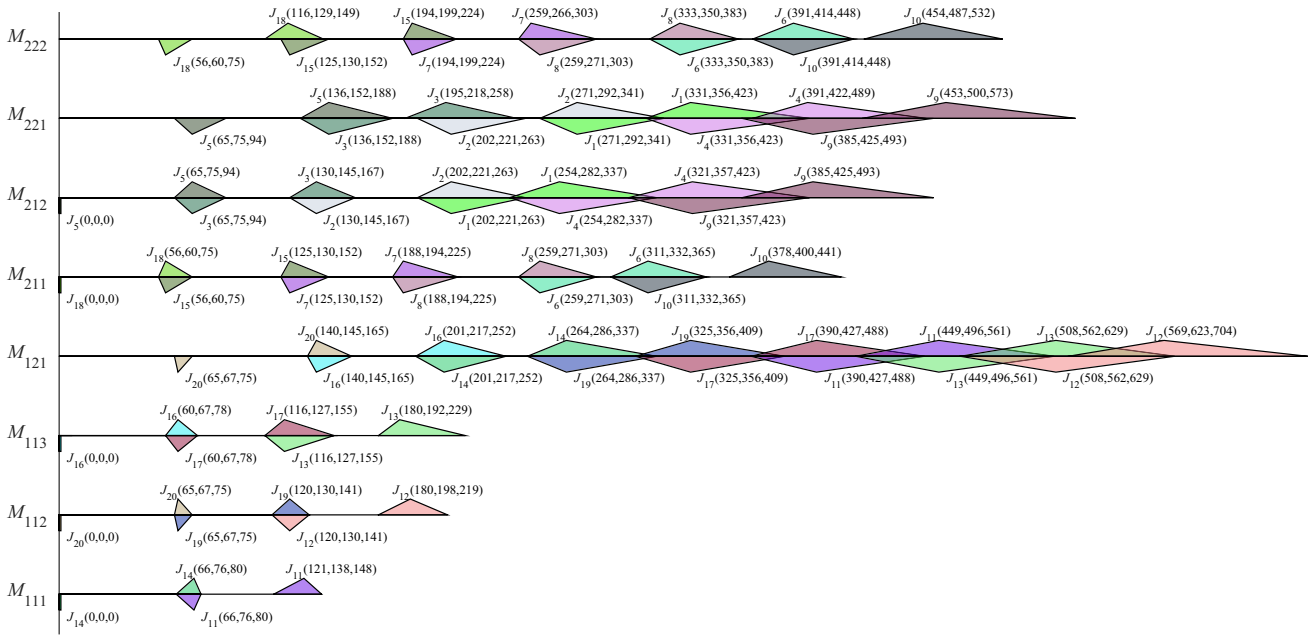


Fig. 2 A schedule of the example

TEC(y) and at least one objective, take TAI as an example, TAI(x) > TAI(y).

Initially, $\bar{P} = P, \bar{s} = s$.

Memeplex evalution

Memeplex evaluation is seldom adopted in the previous SFLAs [6,21,37]. In this study, three quality indices are used to evaluate and discriminate memeplexes.

For memeplex $\mathcal{M}_i$, its quality Me_i is defined by

$$\mathrm{Me}_i = \mathrm{Evo}_i + \mathrm{Sol}_i + \mathrm{Con}_i. \tag{22}$$

Evo_i is defined by

$$\mathrm{Evo}_i = \sum_{x \in \mathcal{M}_i} \bar{\lambda}_x / \lambda_x, \tag{23}$$

where $\bar{\lambda}_x$ and λ_x are the effective search times and search times of $x \in \mathcal{M}_i$. When $x \in \mathcal{M}_i$ is selected as optimization object of memeplex search, $\lambda_x = \lambda_x + 1$ if a new solution is generated, $\bar{\lambda}_x = \bar{\lambda}_x + 1$ if the new solution is not dominated by x, that is, the search of x is effective. $\bar{\lambda}_x$ and λ_x are initialized to 0 before search process is done in each memeplex.

$$\mathrm{Sol}_i = 1 - \sum_{x \in \mathcal{M}_i} \mathrm{rank}_x / \sum_{y \in P} \mathrm{rank}_y, \tag{24}$$

$$\mathrm{Con}_i = |\{x \in \Omega \,|x \in \mathcal{M}_i\}| / |\Omega|, \tag{25}$$

where rank_x is rank of x obtained by non-dominated sorting [9] on population P.

s memeplexes are sorted in the descending order of Me_i. Suppose that $\mathrm{Me}_1 \geq \mathrm{Me}_2 \cdots \geq \mathrm{Me}_s$, so $\mathcal{M}_1$ is the best memeplex and $\mathcal{M}_s$ is the worst one.

Cooperation-based memeplex search

Algorithm 2 describes the main steps of memeplex search. Obviously, memeplex $\mathcal{M}_s$ is just given search times $\eta < \mu$ and $\mathcal{M}_1$ has more search times than μ.

In Algorithm 2, GS, IG and two VNS are used and described in the following. GS(x, u) is described as follows. For a solution $x \in \mathcal{M}_i$, randomly select a non-dominated solution $y \in \mathcal{M}_i$, $y \neq x$; stochastically pick a coding string in the same probability and produce a new z by crossover of the chosen string between x, y; if z is not dominated by x, replace x and update Ω with z and update $\mathcal{Q}$ with x, otherwise, randomly decide a coding string in the same probability and generate a new z by crossover on the selected string between x and a randomly determined $y \in \Omega$; if z is not dominated by x, replace x and update Ω with z and update $\mathcal{Q}$ with x, otherwise randomly produce a solution z; if x keeps invariant, $u = u + 1$, where the set $\mathcal{Q}$ is used to store some intermediate data and can be regarded as memory of CSFLA.

Crossover between x, y is performed in the following way. Randomly decide $k_1, k_2, k_1 < k_2$, decide jobs between k_1 and k_2 on the chosen string of y; if the chosen one is scheduling string, then adjust relative sequence of these jobs on x according to their sequence on y; if the chosen one is factory assignment string, then all genes of x between k_1 and k_2 are replaced with those of y on the same position.

Algorithm 2 Memeplex search

```
for t = 1 to s do
  if t < s then
    η = μ
  else
    η = 2 × μ × Me_s/(Me_1 + Me_s)
  end if
  for v = 1 to η do
    u = 1, randomly select a non-dominated solution x ∈ M_t
    if u = 1 then
      perform GS(x, u)
    end if
    if u = 2 then
      execute VNS1(x, u)
    end if
    if u = 3 then
      apply IG(x, u)
    end if
  end for
  if t = s then
    for v = η + 1 to 2μ do
      randomly choose a non-dominated solution x ∈ M_1
      perform VNS2(x)
    end for
  end if
end for
```

IG(x, u) is shown below. For a solution $x \in \mathcal{M}_i$, a new solution z is first obtained by the following steps: determine one objective randomly, for example, $C_{\max}$, and choose a factory which has the worst objective value; stochastically select jobs π_i, π_j from the chosen factory and delete them from scheduling string, compute the chosen objective value using the remained jobs; reinsert jobs π_i and π_j one by one into a position with the smallest value of the chosen objective among all possible new positions; then if $x \succ z$, update memory $\mathcal{Q}$ and $u = u + 1$; else replace x and update Ω with z.

Three neighborhood structures are used in VNS1 and VNS2. $\mathcal{N}_1$ is shown below. Select a factory f_1 with the biggest completion time and a factory f_2 with the smallest completion time; randomly select a job π_i from factory f_1 and a job π_j from factory f_2 on position k_2 of scheduling string, insert π_i into position $k_2 - 1$ on scheduling string and let $\theta_{\pi_i} = \theta_{\pi_j}$. If $k_2 = 1$, then π_i is deleted and then directly inserted on position 1.

$\mathcal{N}_2$ is described as follows. Decide factories f_1, f_2 as done in $\mathcal{N}_1$, then randomly decide a job π_i from factory f_1 and a job π_j from factory f_2, and exchange them on scheduling string and swap θ_{π_i} and θ_{π_j} on factory assignment string. $\mathcal{N}_3$ is described below. Stochastically choose positions $k_1, k_2, k_3, 1 \leq k_1 < k_2 < k_3 \leq n$, on two strings of a solution, genes between k_1 and $k_2 - 1$ and those between k_2 and k_3 are exchanged.

In IG, $\mathcal{N}_1$ and $\mathcal{N}_2$, factories are not randomly chosen and selected with the worst objective value or the best objective value, the movement of jobs in the chosen factory or between

the selected factories can lead to the improvement of the chosen objective in a high probability, as a result, a new solution can be obtained easily because of the above features, which is non-dominated with the current solution at least.

VNS1(x, u) is performed below. $g = 1, e = 1$, repeat the following steps: produce $z \in \mathcal{N}_g(x)$, if $x \succ z$, then update memory $\mathcal{Q}$ with z and $g = g + 1$; else $g = 1$, replace x and update Ω with z. $e = e + 1$ until $g > 3$ or $e \geq R$.

VNS2(x) is described in Algorithm 3.

Algorithm 3 VNS2(x)

$g = 1, e = 1$
while $g \leq 3$ and $e \leq R$ **do**
  produce $z \in \mathcal{N}_g(x)$
  if $z \succ x$ **then**
    apply non-dominated sorting [9] on $\mathcal{M}_s$,
    build a set $\Theta = \left\{ y \in \mathcal{M}_s \middle| x \succ y, rank_y = \max_{x' \in \mathcal{M}_s} \{rank_{x'}\} \right\}$,
    replace a randomly selected solution $y \in \Theta$ with x,
    replace x and update Ω with z, $g = 1$
  else
    build a set $\left\{ y \in \mathcal{M}_s \middle| z \succ y, rank_y = \max_{x' \in \mathcal{M}_s} \{rank_{x'}\} \right\}$,
    replace a randomly selected solution y from the set with z,
    update memory $\mathcal{Q}$ with z, $g = g + 1$
  end if
  $e = e + 1$
end while

Ω is updated as follows. z is added to Ω and all solutions in Ω are compared based on Pareto dominance, then all dominated solutions are removed. The same way is also used to update $\mathcal{Q}$ with z.

In Algorithm 2, IG, GS and two VNS are used in a flexible way, as a result, their search advantages can be used fully and exploration ability is intensified; on the other hand, $\eta = 2 \times \mu \times \mathrm{Me}_s/(\mathrm{Me}_1 + \mathrm{Me}_s) < \mu$ for $\mathrm{Me}_s < \mathrm{Me}_1$, that is, the search times of $\mathcal{M}_s$ is less than μ, the remained $\mu - \eta$ times are provided for the best memeplex $\mathcal{M}_1$ and VNS2 acts on a non-dominated solution of $\mathcal{M}_1$ and new solutions are used to update $\mathcal{M}_s$. This is an effective cooperation between the worst memeplex $\mathcal{M}_s$ and the best memeplex $\mathcal{M}_1$ by exchanging search times of $\mathcal{M}_s$ and search ability of $\mathcal{M}_1$.

Algorithm description

CSFLA is shown in Algorithm 4. The initial Ω is composed of the non-dominated solutions in initial P. The initial $\mathcal{Q}$ is empty.

Adaptive population shuffling is done in the following way. Let $\bar{P}$ be empty, all memeplexes $\mathcal{M}_i$ meeting $Evo_i \leq \gamma$ are added into $\bar{P}$ and then all solutions $\mathcal{Q}$ are added into $\bar{P}$; finally, non-dominated sorting is performed on all solutions in $\bar{P}$ and the first $N \times \bar{s}/s$ solutions are only kept, where γ is a real number and set to be 0.5 by experiments.

Algorithm 4 CSFLA

randomly produce initial population P with N solutions and and initialize Ω and $\mathcal{Q}$. $\bar{P} = P, \bar{s} = s$.
while the terminal condition is not met **do**
  divide population $\bar{P}$ into $\bar{s}$ memeplexes by binary tournament .
  evaluate the quality of each memeplex and sort all memeplexes according to quality.
  perform searches of all memeplexes.
  population shuffling to form a new $\bar{P}$.
end while
Output Ω.

CSFLA has the following features. (1) Memeplex is evaluated according to solution quality, evolution quality and contribution degree for archive. All memeplexes are sorted in terms of quality. (2) An effective cooperation between the best memeplex and the worst memeplex is implemented. (3)Memeplexes for shuffling are selected in an adaptive way and combined with $\mathcal{Q}$ to form a new population.

Computational experiments

All experiments are implemented using Microsoft Visual C++ 2019 and run on 8.0 G RAM 2.4 GHz CPU PC.

Test instances, comparative algorithms and metrics

Extensive experiments are conducted on 80 instances to test the performances of CSFLA for DEHFSP. 80 instances are randomly produced and the basic information are shown in Table 3. TFN $\widetilde{p}_{iflk} = \left(\delta_1 \times p'_{iflk}, p'_{iflk}, \delta_2 \times p'_{iflk}\right)$, where $p'_{iflk} \in [60, 80]$, $\delta_1 \in [0.85, 1]$ and $\delta_2 \in [1, 1.3]$. Fuzzy due date $\widetilde{d}_i = \left(d'_i, \delta_2 \times d'_i\right)$, where $d'_i \in [(n/F + m - 1) \times 80, m \times 80]$. $w_{fl} \in [1, 3]$. $E_{flk} \in [2, 4]$. $\mathrm{SE}_{flk} = 1$. All the above data are integers except that δ_1 and δ_2 are real number.

Four comparative algorithms are chosen, which are multi-objective tabu search method (MOTS, [48]), multi-objective colonial competitive algorithm (MO-CCA, [14]), multi-objective adaptive large neighborhood search (MOALNS, [40]) and cooperative coevolution algorithm (CCA, [54]). Two variants of CSFLA are also compared with CSFLA to show the effect of new strategies of CSFLA.

MOTS is used to solve two-stage hybrid flow shop with preventive maintenance. It starts from some non-dominated solutions and generates a set of neighborhood solutions for each starting solution. MOTS can be applied to handle DEHFSP after coding string on maintenance is replaced by factory assignment string, neighborhood structure of maintenance is deleted and a neighborhood structure is added, in which a θ_i is randomly selected from factory assignment string and assigned a new value.

Table 3 Information of 80 instances

Instance	n	F	m	Instance	n	F	m	Instance	n	F	m
1	20	2	2	28	40	3	8	55	80	4	6
2	20	2	4	29	60	3	2	56	80	4	8
3	20	2	6	30	60	3	4	57	100	4	2
4	20	2	8	31	60	3	6	58	100	4	4
5	40	2	2	32	60	3	8	59	100	4	6
6	40	2	4	33	80	3	2	60	100	4	8
7	40	2	6	34	80	3	4	61	20	5	2
8	40	2	8	35	80	3	6	62	20	5	4
9	60	2	2	36	80	3	8	63	20	5	6
10	60	2	4	37	100	3	2	64	20	5	8
11	60	2	6	38	100	3	4	65	40	5	2
12	60	2	8	39	100	3	6	66	40	5	4
13	80	2	2	40	100	3	8	67	40	5	6
14	80	2	4	41	20	4	2	68	40	5	8
15	80	2	6	42	20	4	4	69	60	5	2
16	80	2	8	43	20	4	6	70	60	5	4
17	100	2	2	44	20	4	8	71	60	5	6
18	100	2	4	45	40	4	2	72	60	5	8
19	100	2	6	46	40	4	4	73	80	5	2
20	100	2	8	47	40	4	6	74	80	5	4
21	20	3	2	48	40	4	8	75	80	5	6
22	20	3	4	49	60	4	2	76	80	5	8
23	20	3	6	50	60	4	4	77	100	5	2
24	20	3	8	51	60	4	6	78	100	5	4
25	40	3	2	52	60	4	8	79	100	5	6
26	40	3	4	53	80	4	2	80	100	5	8
27	40	3	6	54	80	4	4				

MOCCA is proposed for solving HFSP with the minimization of makespan and total weighted tardiness and has good performance. It can be directly applied to solve DEHFSP after a factory assignment string is added and then VNS1 and GS are used instead of the original VNS and crossover between colony and its imperialist.

MOALNS is dedicated to deal with distributed reentrant permutation flow shop scheduling problem with three objectives. MOALNS can be used to solve DEHFSP after cost and tardiness are replaced with TEC and TAI.

CCA is presented to address multi-objectives DHFSP with fuzzy processing times and some different objectives from our DEHFSP and can be directly used to dispose DEHFSP, so it is chosen as a comparative algorithm.

We construct two variants of CSFLA, which are SFLA1 and SFLA2. When memeplex cooperation is moved from CSFLA, SFLA1 is obtained. The comparison between CSFLA and SFLA1 is to show the effect of cooperation. When shuffling is done in SFLA in the fourth section, SFLA2 is obtained. The comparison between CSFLA and SFLA2 is to reveal the impact of adaptive population shuffling.

Three metrics $\mathcal{C}$ [55], ρ_l [24] and inverted generational distance (IGD, [12,36]) are applied to evaluate the performances of algorithms.

Metric $\mathcal{C}$ is used to measure the dominance relationship between the non-dominated solution sets of L and B. $\mathcal{C}(L, B)$ measures the proportion of members of B which are dominated by that of L.

$$\mathcal{C}(L, B) = \frac{|\{b \in B : \exists l \in L, l \succ b\}|}{|B|}. \tag{26}$$

Metric ρ_l indicates the ratio of number of the elements in the set $\{x \in \Omega_l | x \in \Omega^*\}$ to $|\Omega^*|$. The larger the value of ρ_l, the more members that Ω_l provides for Ω^*. $\rho_l = 1$ indicates that all solutions of Ω^* are provided by Ω_l.

IGD mainly used to measure the distance between the solution set Ω_A of algorithm A and the reference set Ω^*. The

Table 4 Parameters and their levels

Parameters	Factor level			
	1	2	3	4
N	40	50	60	70
s	2	4	6	8
μ	60	80	100	120
R	10	20	30	40

Table 5 The orthogonal array $L_{16}\left(4^4\right)$

Experiment number	Factor level				IGD
	N	s	μ	R	
1	1	1	1	1	0.0814
2	1	2	2	2	0.0557
3	1	3	3	3	0.0168
4	1	4	4	4	0.0845
5	2	1	2	3	0.0927
6	2	2	1	4	0.0970
7	2	3	4	1	0.0174
8	2	4	3	2	0.0487
9	3	1	3	4	0.0099
10	3	2	4	3	0.0717
11	3	3	1	2	0.0129
12	3	4	2	1	0.0137
13	4	1	4	2	0.0105
14	4	2	3	1	0.0548
15	4	3	2	4	0.0255
16	4	4	1	3	0.0988

smaller the value of IGD, the better the overall performance of algorithm A.

$$\mathrm{IGD}(\Omega_A, \Omega^*) = \frac{1}{|\Omega^*|} \sum_{x \in \Omega^*} \min_{y \in \Omega_A} d(x, y), \tag{27}$$

where $d(x, y)$ is the Euclidean distance between solution x and y by normalized objectives.

Parameter settings

CSFLA has five main parameters: stopping condition, N, s, μ, R. We set $0.1 \times n \times m$ CPU time as stopping condition. We found that CSFLA, its two variants and four comparative algorithms can converge well when the above time reaches, so we first choose $0.1 \times n \times m$ CPU time as stopping condition.

Taguchi method is used to decide settings of other parameters. Instances 1, 50 and 80 are selected. The same settings can be obtained using these instances, so we only exhibit the results on instance 50.

Table 4 lists the levels of each parameter and Table 5 gives the orthogonal array $L_9\left(3^3\right)$. CSFLA runs 10 times independently for instance 50. The results of IGD and S/N ratio are shown in Fig. 3, in which S/N ratio is defined as $-10 \times \log_{10}\left(\mathrm{IGD}^2\right)$. As shown in Fig. 3, the best setting is $N = 60$, $s = 6$, $\mu = 100$ and $R = 20$.

All parameters of MOTS, MOCCA, MOALNS and CCA are directly selected from Wang and Liu [48], Karimi and Davoudpour [14], Rifai et al. [40] and Zheng et al. [54] except the stopping condition. Parameters of SFLA1 and SFLA2 are identical with those of CSFLA.

Results and analyses

Each algorithm randomly runs 10 times for each instance. The reference set Ω^* is composed of the non-dominated solutions in $\bigcup_{i=1}^{6} \Omega_i$, where Ω_1, Ω_2, Ω_3, Ω_4, Ω_5 and Ω_6 are non-dominated solutions of CSFLA, MOTS, MOCCA, MOALNS, SFLA1 and SFLA2 obtained in 10 runs. Tables 6, 7, 8 and 9 report the results of all algorithms, in which symbol 'Ins' indicates instance and 'CS', 'MT', 'MC', 'MA', 'C', 'S1' and 'S2' represent CSFLA, MOTS, MOCCA, MOALNS, CCA, SFLA1 and SFLA2. Figure 4 exhibits box plot of all algorithms on ρ, $\mathcal{C}$ and IGD. Figure 5 provides the distribution of non-dominated solutions of algorithms on instances 32 and 68, in which $c_1(C_{\max})$, $c_1(\mathrm{TEC})$ and TAI are used.

As shown in Tables 6, 8 and 9, CSFLA obtains smaller IGD than SFLA1 on 79 instances and it also has smaller $\mathcal{C}(\mathrm{S1}, \mathrm{CS})$ than $\mathcal{C}(\mathrm{CS}, \mathrm{S1})$ on 79 instances; moreover, $\mathcal{C}(\mathrm{CS}, \mathrm{S1})$ is equal to 1 on 49 instances, that is, all solutions of SFLA1 is dominated by non-dominated solutions of CSFLA. Similarly, ρ of CSFLA is more than that of SFLA1 on all instances and equal to 1 on 14 instances, that is, all members of the reference set Ω^* are provided by CSFLA. CSLFA performs notably better than SFLA1. This conclusion also can be obtained from Figs. 4 and 5. The same conclusion can be obtained by comparing CSFLA with SFLA2. The notable performance differences between CSFLA and its two variants reveal that cooperation and adaptive population shuffling really have positive impact on the performances of CSFLA.

As illustrated in Tables 7 and 9, solutions of MOTS are always far away from those of CSFLA on 79 instances because IGD of CSFLA is less than that of MOTS and $\mathcal{C}(\mathrm{CS}, \mathrm{MT})$ is equal to 1 on 45 instances; CSFLA produces smaller IGD than MOCCA and obtains bigger $\mathcal{C}(\mathrm{CS}, \mathrm{MC})$ than $\mathcal{C}(\mathrm{MC}, \mathrm{CS})$ on all instances;

IGD of CSFLA is smaller than CCA on 68 instance and $\mathcal{C}(\mathrm{C}, \mathrm{CS})$ is less than $\mathcal{C}(\mathrm{CS}, \mathrm{C})$ on 71 instances, and it can be seen that MOALNS performs significantly worse than CSFLA on metrics IGD and $\mathcal{C}$ on all instances. On the other hand, ρ of CSFLA is more than that of its all comparative algorithms on 79 instances, that is, most of the solutions of

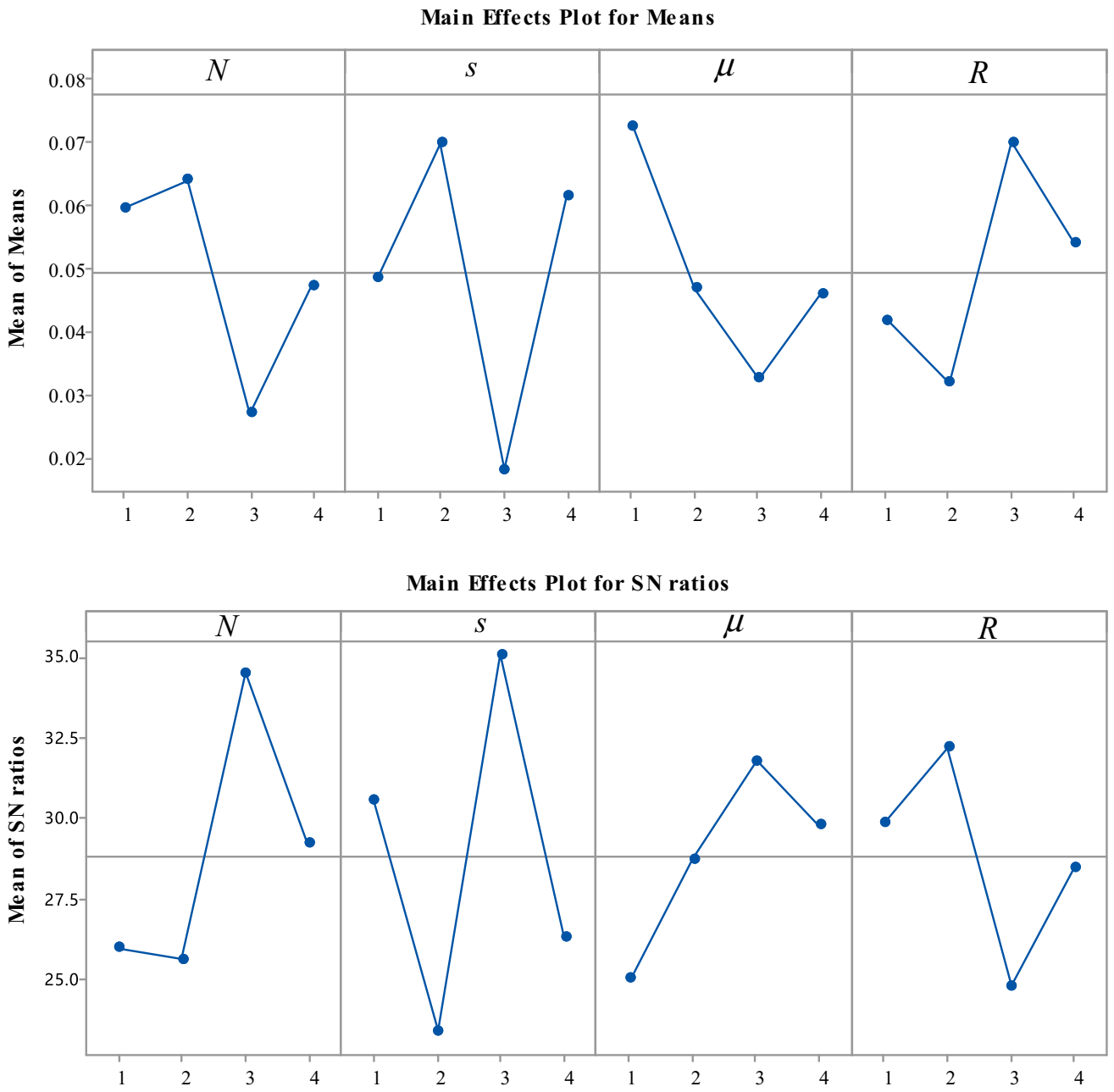


Fig. 3 Main effects plot for means and main effects plot for S/N ratios

Ω^* are provided by CSFLA. Obviously, CSLFA performs notably better than its four comparative algorithms. The same conclusion also can be drawn from box plot in Figs. 4 and 5.

The promising performances of CSFLA mainly result from its cooperation between memeplexes and adaptive shuffling. With the addition of cooperation, the waste of computing resource on the worst memeplex is avoided and the good search ability of the best memeplex is used fully. Adaptive shuffling can keep the stability of solution structure in some memeplexes with good evolution quality. These strategies can effectively keep high diversity and avoid search falling local optima; thus, it can be concluded that CSFLA is a competitive method for solving DEHFSP with fuzzy processing time.

Conclusion

In real-world manufacturing systems, energy consumption, uncertainty and multi-objective optimization are often required to be considered simultaneously. This paper aims to solve DEHFSP with fuzzy processing time and apply a new

Table 6 Comparison results of CSFLA, SFLA1 and SFLA2 on metric $\mathcal{C}$

Ins	$\mathcal{C}$(CS,S1)	$\mathcal{C}$(S1,CS)	$\mathcal{C}$(CS,S2)	$\mathcal{C}$(S2,CS)	Ins	$\mathcal{C}$(CS,S1)	$\mathcal{C}$(S1,CS)	$\mathcal{C}$(CS,S2)	$\mathcal{C}$(S2,CS)
1	1.000	0.000	1.000	0.000	41	0.978	0.000	1.000	0.000
2	0.976	0.000	1.000	0.000	42	1.000	0.000	1.000	0.000
3	1.000	0.000	1.000	0.000	43	1.000	0.000	0.932	0.000
4	0.891	0.011	0.984	0.006	44	0.993	0.000	0.981	0.000
5	1.000	0.000	1.000	0.000	45	1.000	0.000	1.000	0.000
6	1.000	0.000	1.000	0.000	46	0.881	0.004	0.989	0.000
7	1.000	0.000	1.000	0.000	47	1.000	0.000	1.000	0.000
8	1.000	0.000	1.000	0.000	48	1.000	0.000	1.000	0.000
9	1.000	0.000	1.000	0.000	49	0.879	0.086	0.823	0.019
10	0.872	0.000	0.920	0.000	50	1.000	0.000	1.000	0.000
11	1.000	0.000	1.000	0.000	51	1.000	0.000	1.000	0.000
12	1.000	0.000	1.000	0.000	52	1.000	0.000	1.000	0.000
13	0.561	0.000	0.971	0.000	53	0.895	0.000	0.597	0.005
14	1.000	0.000	1.000	0.000	54	0.982	0.000	1.000	0.000
15	1.000	0.000	1.000	0.000	55	1.000	0.000	1.000	0.000
16	1.000	0.000	1.000	0.000	56	1.000	0.000	1.000	0.000
17	0.636	0.000	1.000	0.000	57	0.959	0.000	0.753	0.014
18	0.400	0.023	0.797	0.023	58	1.000	0.000	1.000	0.000
19	1.000	0.000	1.000	0.000	59	1.000	0.000	0.986	0.000
20	1.000	0.000	0.979	0.000	60	1.000	0.000	1.000	0.000
21	0.830	0.011	0.870	0.026	61	1.000	0.000	1.000	0.000
22	1.000	0.000	0.989	0.006	62	0.757	0.417	0.989	0.000
23	0.986	0.000	0.992	0.000	63	0.957	0.024	0.969	0.024
24	0.184	0.485	1.000	0.000	64	1.000	0.000	0.912	0.000
25	0.958	0.000	0.974	0.000	65	1.000	0.000	0.986	0.000
26	1.000	0.000	0.906	0.000	66	1.000	0.000	0.763	0.022
27	0.979	0.000	1.000	0.000	67	1.000	0.000	1.000	0.000
28	0.984	0.000	1.000	0.000	68	1.000	0.000	1.000	0.000
29	1.000	0.000	1.000	0.000	69	0.913	0.000	0.804	0.014
30	1.000	0.000	1.000	0.000	70	0.947	0.000	0.935	0.000
31	1.000	0.000	1.000	0.000	71	1.000	0.000	1.000	0.000
32	1.000	0.000	1.000	0.000	72	1.000	0.000	1.000	0.000
33	1.000	0.000	1.000	0.000	73	0.864	0.000	0.840	0.000
34	0.971	0.000	1.000	0.000	74	1.000	0.000	1.000	0.000
35	1.000	0.000	1.000	0.000	75	1.000	0.000	1.000	0.000
36	0.992	0.000	1.000	0.000	76	0.925	0.000	0.899	0.000
37	0.993	0.000	0.977	0.000	77	1.000	0.000	0.891	0.000
38	1.000	0.000	1.000	0.000	78	0.956	0.000	0.976	0.000
39	0.778	0.000	1.000	0.000	79	1.000	0.000	1.000	0.000
40	0.960	0.000	1.000	0.000	80	1.000	0.000	1.000	0.000

Table 7 Comparison results of CSFLA, SFLA1 and SFLA2 on metric $\mathcal{C}$

Ins	$\mathcal{C}$(CS,MT)	$\mathcal{C}$(MT,CS)	$\mathcal{C}$(CS,MC)	$\mathcal{C}$(MC,CS)	$\mathcal{C}$(CS,MA)	$\mathcal{C}$(MA,CS)	$\mathcal{C}$(CS,C)	$\mathcal{C}$(C,CS)
1	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
2	0.940	0.033	1.000	0.000	1.000	0.000	0.900	0.021
3	0.968	0.000	1.000	0.000	1.000	0.000	1.000	0.000
4	0.620	0.050	1.000	0.000	1.000	0.000	1.000	0.000
5	1.000	0.000	1.000	0.000	1.000	0.000	0.714	0.014
6	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
7	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
8	0.994	0.000	1.000	0.000	1.000	0.000	0.462	0.015
9	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
10	0.976	0.000	1.000	0.000	1.000	0.000	0.167	0.078
11	0.992	0.000	1.000	0.000	1.000	0.000	0.556	0.002
12	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
13	0.905	0.000	1.000	0.000	1.000	0.000	0.778	0.049
14	1.000	0.000	1.000	0.000	1.000	0.000	0.600	0.017
15	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
16	1.000	0.000	1.000	0.000	1.000	0.000	0.500	0.024
17	1.000	0.000	1.000	0.000	1.000	0.000	0.000	1.000
18	0.864	0.023	1.000	0.000	1.000	0.000	0.167	0.138
19	1.000	0.000	1.000	0.000	1.000	0.000	0.667	0.000
20	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.000
21	0.817	0.079	1.000	0.000	1.000	0.000	0.905	0.032
22	0.885	0.025	1.000	0.000	1.000	0.000	1.000	0.000
23	0.453	0.211	1.000	0.000	1.000	0.000	1.000	0.000
24	0.448	0.309	0.941	0.000	1.000	0.000	1.000	0.000
25	1.000	0.000	1.000	0.000	1.000	0.000	0.947	0.000
26	1.000	0.000	1.000	0.000	1.000	0.000	0.545	0.000
27	0.865	0.012	1.000	0.000	1.000	0.000	1.000	0.000
28	0.792	0.017	1.000	0.000	1.000	0.000	1.000	0.000
29	1.000	0.000	1.000	0.000	1.000	0.000	0.667	0.029
30	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
31	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
32	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
33	1.000	0.000	1.000	0.000	1.000	0.000	0.091	0.332
34	1.000	0.000	1.000	0.000	1.000	0.000	0.400	0.058
35	1.000	0.000	1.000	0.000	1.000	0.000	0.750	0.000
36	1.000	0.000	1.000	0.000	1.000	0.000	0.750	0.000
37	0.993	0.000	1.000	0.000	1.000	0.000	0.000	0.482
38	1.000	0.000	1.000	0.000	1.000	0.000	0.750	0.004
39	0.923	0.000	1.000	0.000	1.000	0.000	0.000	0.162
40	1.000	0.000	1.000	0.000	1.000	0.000	0.333	0.000

Table 7 continued

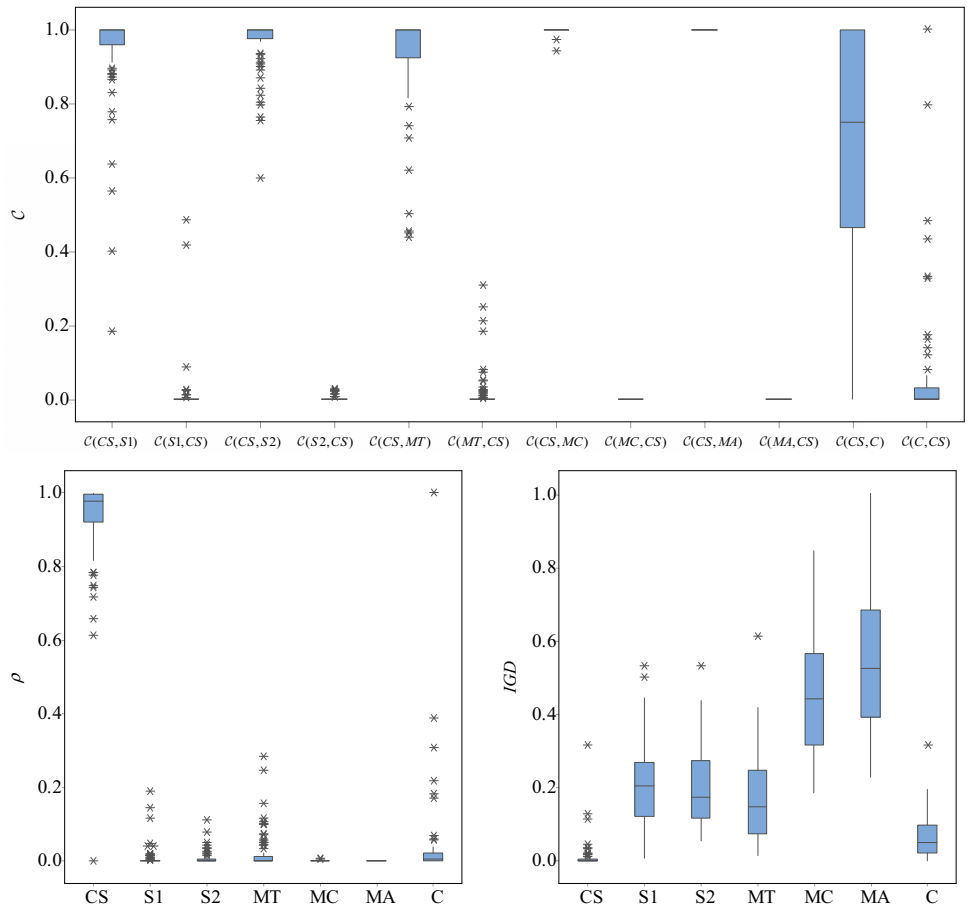
Ins	$\mathcal{C}$(CS,MT)	$\mathcal{C}$(MT,CS)	$\mathcal{C}$(CS,MC)	$\mathcal{C}$(MC,CS)	$\mathcal{C}$(CS,MA)	$\mathcal{C}$(MA,CS)	$\mathcal{C}$(CS,C)	$\mathcal{C}$(C,CS)
41	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
42	0.500	0.249	1.000	0.000	1.000	0.000	0.500	0.061
43	0.437	0.182	1.000	0.000	1.000	0.000	1.000	0.000
44	1.000	0.000	1.000	0.000	1.000	0.000	0.600	0.000
45	0.996	0.004	1.000	0.000	1.000	0.000	0.474	0.059
46	0.833	0.019	1.000	0.000	1.000	0.000	0.600	0.061
47	0.973	0.002	1.000	0.000	1.000	0.000	0.833	0.000
48	0.995	0.000	1.000	0.000	1.000	0.000	1.000	0.000
49	0.967	0.010	1.000	0.000	1.000	0.000	0.000	0.432
50	0.972	0.000	1.000	0.000	1.000	0.000	1.000	0.000
51	1.000	0.000	1.000	0.000	1.000	0.000	0.900	0.003
52	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
53	0.897	0.000	1.000	0.000	1.000	0.000	0.692	0.030
54	1.000	0.000	1.000	0.000	1.000	0.000	0.667	0.000
55	1.000	0.000	1.000	0.000	1.000	0.000	0.857	0.010
56	1.000	0.000	1.000	0.000	1.000	0.000	0.333	0.042
57	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.796
58	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.327
59	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.040
60	1.000	0.000	1.000	0.000	1.000	0.000	0.000	0.174
61	1.000	0.000	1.000	0.000	1.000	0.000	0.857	0.000
62	0.910	0.009	1.000	0.000	1.000	0.000	0.556	0.014
63	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
64	0.873	0.073	0.974	0.000	1.000	0.000	0.857	0.000
65	0.885	0.047	1.000	0.000	1.000	0.000	0.250	0.000
66	0.988	0.000	1.000	0.000	1.000	0.000	1.000	0.000
67	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
68	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
69	0.984	0.000	1.000	0.000	1.000	0.000	0.846	0.005
70	0.739	0.000	1.000	0.000	1.000	0.000	0.375	0.026
71	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
72	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
73	0.707	0.015	1.000	0.000	1.000	0.000	0.444	0.028
74	1.000	0.000	1.000	0.000	1.000	0.000	0.750	0.004
75	0.873	0.000	1.000	0.000	1.000	0.000	0.600	0.005
76	0.944	0.000	1.000	0.000	1.000	0.000	0.715	0.000
77	0.931	0.000	1.000	0.000	1.000	0.000	0.947	0.000
78	1.000	0.000	1.000	0.000	1.000	0.000	0.801	0.000
79	1.000	0.000	1.000	0.000	1.000	0.000	0.532	0.064
80	1.000	0.000	1.000	0.000	1.000	0.000	0.259	0.120

Table 8 Comparison results of CSFLA, SFLA1, SFLA2, MOTS, MOCCA, MOALNS, CCA on metric ρ

Ins	CS	S1	S2	MT	MC	MA	C	Ins	CS	S1	S2	MT	MC	MA	C
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	41	0.991	0.009	0.000	0.000	0.000	0.000	0.000
2	0.942	0.012	0.000	0.041	0.000	0.000	0.004	42	0.817	0.000	0.000	0.156	0.000	0.000	0.028
3	0.990	0.000	0.000	0.010	0.000	0.000	0.000	43	0.743	0.000	0.011	0.246	0.000	0.000	0.000
4	0.885	0.013	0.003	0.099	0.000	0.000	0.000	44	0.988	0.003	0.003	0.000	0.000	0.000	0.006
5	0.993	0.000	0.000	0.000	0.000	0.000	0.007	45	0.958	0.000	0.000	0.004	0.000	0.000	0.038
6	1.000	0.000	0.000	0.000	0.000	0.000	0.000	46	0.935	0.000	0.000	0.050	0.000	0.000	0.015
7	1.000	0.000	0.000	0.000	0.000	0.000	0.000	47	0.993	0.000	0.000	0.005	0.000	0.000	0.002
8	0.990	0.000	0.000	0.000	0.000	0.000	0.010	48	0.998	0.000	0.000	0.002	0.000	0.000	0.000
9	1.000	0.000	0.000	0.000	0.000	0.000	0.000	49	0.927	0.000	0.005	0.000	0.000	0.000	0.067
10	0.922	0.039	0.000	0.007	0.000	0.000	0.033	50	0.987	0.000	0.000	0.013	0.000	0.000	0.000
11	0.992	0.000	0.000	0.000	0.000	0.000	0.008	51	0.998	0.000	0.000	0.000	0.000	0.000	0.002
12	1.000	0.000	0.000	0.000	0.000	0.000	0.000	52	1.000	0.000	0.000	0.000	0.000	0.000	0.000
13	0.819	0.145	0.006	0.006	0.000	0.000	0.024	53	0.855	0.009	0.110	0.009	0.000	0.000	0.018
14	0.994	0.000	0.000	0.000	0.000	0.000	0.006	54	0.992	0.005	0.000	0.000	0.000	0.000	0.003
15	1.000	0.000	0.000	0.000	0.000	0.000	0.000	55	0.998	0.000	0.000	0.000	0.000	0.000	0.002
16	0.995	0.000	0.000	0.000	0.000	0.000	0.005	56	0.987	0.000	0.000	0.000	0.000	0.000	0.013
17	0.000	0.000	0.000	0.000	0.000	0.000	1.000	57	0.777	0.000	0.043	0.000	0.000	0.000	0.181
18	0.953	0.000	0.009	0.017	0.000	0.000	0.021	58	0.971	0.000	0.000	0.000	0.000	0.000	0.029
19	0.995	0.000	0.000	0.000	0.000	0.000	0.005	59	0.978	0.000	0.004	0.000	0.000	0.000	0.018
20	0.981	0.000	0.003	0.000	0.000	0.000	0.016	60	0.991	0.000	0.000	0.000	0.000	0.000	0.009
21	0.833	0.039	0.025	0.098	0.000	0.000	0.005	61	0.986	0.000	0.000	0.000	0.000	0.000	0.014
22	0.923	0.000	0.006	0.071	0.000	0.000	0.000	62	0.783	0.115	0.006	0.070	0.000	0.000	0.025
23	0.717	0.000	0.000	0.283	0.000	0.000	0.000	63	0.930	0.047	0.023	0.000	0.000	0.000	0.000
24	0.747	0.188	0.000	0.059	0.005	0.000	0.000	64	0.857	0.000	0.034	0.101	0.004	0.000	0.004
25	0.977	0.014	0.005	0.000	0.000	0.000	0.005	65	0.870	0.000	0.006	0.105	0.000	0.000	0.019
26	0.972	0.000	0.017	0.000	0.000	0.000	0.011	66	0.924	0.000	0.076	0.000	0.000	0.000	0.000
27	0.955	0.000	0.000	0.045	0.000	0.000	0.000	67	1.000	0.000	0.000	0.000	0.000	0.000	0.000
28	0.966	0.002	0.000	0.032	0.000	0.000	0.000	68	1.000	0.000	0.000	0.000	0.000	0.000	0.000
29	0.981	0.000	0.000	0.000	0.000	0.000	0.019	69	0.930	0.013	0.048	0.000	0.000	0.000	0.009
30	1.000	0.000	0.000	0.000	0.000	0.000	0.000	70	0.870	0.002	0.002	0.116	0.000	0.000	0.010
31	1.000	0.000	0.000	0.000	0.000	0.000	0.000	71	1.000	0.000	0.000	0.000	0.000	0.000	0.000
32	1.000	0.000	0.000	0.000	0.000	0.000	0.000	72	1.000	0.000	0.000	0.000	0.000	0.000	0.000
33	0.944	0.000	0.000	0.000	0.000	0.000	0.056	73	0.894	0.017	0.006	0.055	0.000	0.000	0.029
34	0.963	0.015	0.000	0.000	0.000	0.000	0.022	74	0.996	0.000	0.000	0.000	0.000	0.000	0.004
35	0.997	0.000	0.000	0.000	0.000	0.000	0.003	75	0.947	0.000	0.000	0.048	0.000	0.000	0.005
36	0.997	0.000	0.000	0.000	0.000	0.000	0.003	76	0.830	0.000	0.000	0.000	0.000	0.000	0.170
37	0.926	0.000	0.012	0.003	0.000	0.000	0.058	77	0.922	0.000	0.034	0.022	0.000	0.000	0.022
38	0.996	0.000	0.000	0.000	0.000	0.000	0.004	78	0.656	0.018	0.018	0.000	0.000	0.000	0.307
39	0.966	0.000	0.000	0.000	0.000	0.000	0.034	79	0.613	0.000	0.000	0.000	0.000	0.000	0.387
40	0.996	0.000	0.000	0.000	0.000	0.000	0.004	80	0.783	0.000	0.000	0.000	0.000	0.000	0.217

Table 9 Comparison results of CSFLA, SFLA1, SFLA2, MOTS, MOCCA, MOALNS, CCA on metric IGD

Ins	CS	S1	S2	MT	MC	MA	C	Ins	CS	S1	S2	MT	MC	MA	C
1	0.000	0.172	0.117	0.109	0.337	0.510	0.150	41	0.000	0.090	0.076	0.053	0.234	0.512	0.068
2	0.003	0.090	0.080	0.052	0.232	0.316	0.040	42	0.035	0.094	0.102	0.025	0.273	0.272	0.035
3	0.000	0.134	0.145	0.060	0.415	0.621	0.102	43	0.005	0.120	0.113	0.016	0.252	0.337	0.163
4	0.001	0.051	0.082	0.016	0.286	0.351	0.137	44	0.000	0.103	0.068	0.075	0.206	0.244	0.058
5	0.000	0.211	0.253	0.270	0.459	0.514	0.045	45	0.003	0.180	0.167	0.140	0.380	0.440	0.053
6	0.000	0.227	0.210	0.243	0.497	0.524	0.038	46	0.005	0.142	0.119	0.073	0.225	0.284	0.026
7	0.000	0.344	0.357	0.304	0.658	0.794	0.317	47	0.000	0.152	0.190	0.051	0.383	0.506	0.062
8	0.000	0.154	0.195	0.129	0.343	0.390	0.020	48	0.000	0.192	0.158	0.083	0.317	0.357	0.096
9	0.000	0.337	0.437	0.392	0.848	1.005	0.118	49	0.044	0.228	0.192	0.190	0.362	0.444	0.000
10	0.004	0.088	0.111	0.133	0.360	0.462	0.002	50	0.000	0.151	0.195	0.118	0.402	0.597	0.080
11	0.000	0.344	0.411	0.327	0.526	0.569	0.025	51	0.000	0.234	0.252	0.218	0.432	0.528	0.046
12	0.000	0.500	0.364	0.331	0.732	0.785	0.125	52	0.000	0.304	0.323	0.115	0.707	0.793	0.104
13	0.002	0.051	0.103	0.126	0.229	0.254	0.047	53	0.002	0.131	0.056	0.095	0.468	0.572	0.057
14	0.000	0.335	0.364	0.359	0.560	0.626	0.036	54	0.000	0.204	0.209	0.216	0.544	0.676	0.032
15	0.000	0.399	0.299	0.296	0.533	0.781	0.050	55	0.000	0.219	0.235	0.181	0.570	0.690	0.042
16	0.001	0.446	0.439	0.368	0.690	0.748	0.016	56	0.002	0.409	0.421	0.324	0.784	0.928	0.015
17	0.316	0.534	0.533	0.613	0.820	0.842	0.000	57	0.129	0.285	0.309	0.280	0.185	0.228	0.000
18	0.011	0.149	0.153	0.164	0.239	0.451	0.011	58	0.022	0.208	0.203	0.258	0.591	0.740	0.000
19	0.000	0.248	0.270	0.197	0.602	0.853	0.018	59	0.005	0.252	0.261	0.224	0.443	0.434	0.000
20	0.000	0.266	0.215	0.192	0.395	0.501	0.000	60	0.018	0.321	0.380	0.249	0.593	0.625	0.000
21	0.004	0.057	0.077	0.045	0.247	0.350	0.095	61	0.000	0.105	0.106	0.131	0.262	0.379	0.076
22	0.001	0.098	0.120	0.050	0.316	0.403	0.077	62	0.032	0.057	0.104	0.051	0.245	0.275	0.045
23	0.006	0.109	0.118	0.018	0.292	0.332	0.073	63	0.003	0.111	0.148	0.198	0.417	0.443	0.137
24	0.017	0.008	0.136	0.031	0.373	0.431	0.175	64	0.003	0.193	0.133	0.073	0.260	0.330	0.137
25	0.000	0.145	0.091	0.143	0.312	0.388	0.104	65	0.001	0.133	0.124	0.049	0.469	0.573	0.018
26	0.000	0.118	0.085	0.093	0.261	0.304	0.022	66	0.001	0.140	0.085	0.081	0.485	0.596	0.099
27	0.000	0.137	0.174	0.053	0.460	0.570	0.051	67	0.000	0.253	0.161	0.095	0.450	0.545	0.195
28	0.000	0.126	0.145	0.031	0.375	0.482	0.060	68	0.000	0.107	0.177	0.106	0.507	0.487	0.064
29	0.004	0.385	0.304	0.414	0.714	0.855	0.164	69	0.001	0.090	0.064	0.107	0.428	0.441	0.098
30	0.000	0.221	0.169	0.150	0.519	0.627	0.060	70	0.001	0.124	0.112	0.055	0.265	0.380	0.019
31	0.000	0.288	0.326	0.182	0.423	0.730	0.083	71	0.000	0.237	0.154	0.101	0.590	0.716	0.074
32	0.000	0.416	0.430	0.420	0.793	0.862	0.155	72	0.000	0.239	0.245	0.130	0.583	0.794	0.116
33	0.018	0.204	0.170	0.182	0.459	0.525	0.005	73	0.004	0.120	0.153	0.144	0.320	0.315	0.025
34	0.004	0.240	0.264	0.311	0.366	0.463	0.037	74	0.000	0.234	0.164	0.168	0.584	0.921	0.051
35	0.000	0.261	0.167	0.248	0.537	0.584	0.022	75	0.000	0.165	0.108	0.063	0.587	0.628	0.022
36	0.000	0.290	0.328	0.291	0.552	0.560	0.118	76	0.000	0.221	0.175	0.162	0.444	0.529	0.059
37	0.113	0.212	0.273	0.203	0.219	0.284	0.000	77	0.000	0.105	0.080	0.068	0.456	0.697	0.058
38	0.000	0.344	0.321	0.310	0.706	0.841	0.053	78	0.000	0.194	0.187	0.167	0.362	0.406	0.107
39	0.007	0.281	0.274	0.275	0.616	0.839	0.000	79	0.001	0.271	0.225	0.226	0.577	0.643	0.036
40	0.000	0.333	0.310	0.312	0.536	0.567	0.033	80	0.010	0.232	0.282	0.176	0.547	0.614	0.016



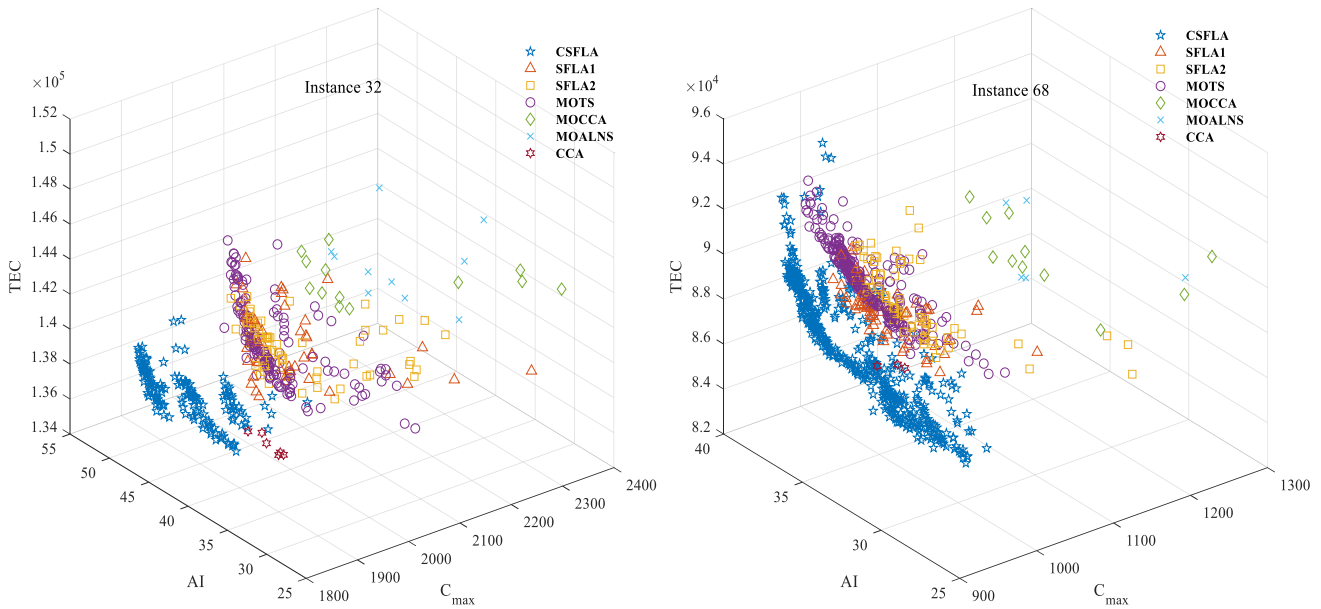


Fig. 4 Box plot of metric ρ, $\mathcal{C}$ and IGD

Fig. 5 The distribution of non-dominated solutions

algorithm named CSFLA to optimize fuzzy makespan, total agreement index and fuzzy total energy consumption simultaneously. In SFLA, three search strategies named IG, GS and VNS are designed based on problem-related features. After memeplexes evaluation, the best memeplex and the worst memeplex cooperate each other through exchanging search times and search ability. An adaptive shuffling is adopted to improve search efficiency. The performances of CSFLA are tested and the computational results show that CSFLA is a promising method to solve the considered DEHFSP.

In the near future, we will continue to study scheduling problems of distributed hybrid flow shop. Swarm intelligence optimizations are also the focus of our attention and we will try to carry out effective labor division and collaboration among multiple groups. We will also pay attention to DHFSP with some special constraints like distributed assembly hybrid flow shop scheduling problem.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant nos. 61573264, 71471151).

References

1. Abdullah S, Abdolrazzagh-Nezhad M (2014) Fuzzy job-shop scheduling problems: a review. Inf Sci 278:380–407
2. Behnamian J (2014) Decomposition based hybrid VNS-TS algorithm for distributed parallel factories scheduling with virtual corporation. Comput Oper Res 52:181–191
3. Behnamian J, Ghomi SMTF (2013) The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. Inf Sci 219:181–196
4. Behnamian J, Ghomi SMTF (2016) A survey of multi-factory scheduling. J Intell Manuf 27:231–249
5. Cai J, Lei D, Li M (2020a) A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop. Int J Prod Res 2020:1–8
6. Cai J, Zhou R, Lei D (2020b) Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. Eng Appl Artif Intel 90:103540
7. Cai J, Zhou R, Lei D (2020c) Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: collaborative variable search. J Intell Fuzzy Syst 38:3189–3199
8. Dai M, Tang D, Zheng K, Cai Q (2013) An improved genetic-simulated annealing algorithm based on a hormone modulation mechanism for a flexible flow-shop scheduling problem. Adv Mech Eng 5:124903
9. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE T Evolut Comput 6:182–197
10. Eusuff M, Lansey K, Pasha F (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optimiz 38:129–154
11. Gao Z, Peng J, Han Z, Jia M (2019) Flow shop scheduling with variable processing times based on differential shuffled frog leaping algorithm. Int J Model Ident Control 33:179–187
12. Gong G, Chiong R, Deng Q, Han W, Zhang L, Lin W, Li K (2020) Energy-efficient flexible flow shop scheduling with worker flexibility. Expert Syst Appl 141:112902
13. Hao JH, Li JQ, Du Y, Song MX, Duan P, Zhang YY (2019) Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm. IEEE Access 7:66879–66894
14. Karimi N, Davoudpour H (2016) Multi-objective colonial competitive algorithm for hybrid flowshop problem. Appl Soft Comput 49:725–733
15. Karpagam M, Geetha K, Rajan C (2020) A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques. Soft Comput 24:637–646
16. Lei D (2010) A genetic algorithm for flexible job shop scheduling with fuzzy processing time. Int J Prod Res 48:2995–3013
17. Lei D, Gao L, Zheng Y (2018) A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. IEEE T Eng Manage 65:330–340
18. Lei D, Guo X (2012) Swarm-based neighbourhood search algorithm for fuzzy flexible job shop scheduling. Int J Prod Res 50:1639–1649
19. Lei D, Liu M (2020) An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. Comput Ind Eng 141:106320
20. Lei D, Wang T (2020) Solving distributed two-stage hybrid flow-shop scheduling using a shuffled frog-leaping algorithm with memeplex grouping. Eng Optimiz 2020:1–14
21. Lei D, Yuan Y, Cai JC (2020) An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. Int J Prod Res 2020:1–13
22. Lei D, Zheng Y, Guo X (2017) A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. Int J Prod Res 55:3126–3140
23. Lei DM, Guo XP (2015) A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents. Expert Syst Appl 42:9333–9339
24. Lei DM, Li M, Wang L (2019) A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. IEEE T Cybern 49:1097–1109
25. Li JQ, Pan QK (2013) Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities. Int J Prod Econ 145:4–17
26. Li JQ, Sang HY, Han YY, Wang CG, Gao KZ (2018) Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. J Clean Prod 181:584–598
27. Li M, Lei D, Cai J (2019) Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. Swarm Evol Comput 49:34–43
28. Li Y, Li X, Gao L, Meng L (2020a) An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times. Comput Ind Eng 2020:147
29. Li Y, Li X, Gao L, Zhang B, Pan QK, Tasgetiren MF, Meng L (2020b) A discrete artificial bee colony algorithm for distributed

hybrid flowshop scheduling problem with sequence-dependent setup times. Int J Prod Res 2020:1–20
30. Lin J (2019) Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time. Eng Appl Artif Intel 77:186–196
31. Lin J, Zhu L, Wang ZJ (2019) A hybrid multi-verse optimization for the fuzzy flexible job-shop scheduling problem. Comput Ind Eng 127:1089–1100
32. Lin W, Yu DY, Zhang C, Liu X, Zhang S, Tian Y, Liu S, Xie Z (2015) A multi-objective teaching-learning-based optimization algorithm to scheduling in turning processes for minimizing makespan and carbon footprint. J Clean Prod 101:337–347
33. Liu GS, Zhou Y, Yang HD (2017) Minimizing energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time. J Clean Prod 147:470–484
34. Luo H, Du B, Huang GQ, Chen H, Li X (2013) Hybrid flow shop scheduling considering machine electricity consumption cost. Int J Prod Econ 146:423–439
35. Meng L, Zhang C, Shao X, Ren Y, Ren C (2019) Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. Int J Prod Res 57:1119–1145
36. Ngoc Hoang L, La Poutre H, Bosman PAN (2018) Multi-objective gene-pool optimal mixing evolutionary algorithm with the interleaved multi-start scheme. Swarm Evol Comput 40:238–254
37. Pan QK, Wang L, Gao L, Li J (2011) An effective shuffled frog-leaping algorithm for lot-streaming flow shop scheduling problem. Int J Adv Manuf Tech 52:699–713
38. Rahimi-Vahed A, Dangchi M, Rafiei H, Salimi E (2009) A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem. Int J Adv Manuf Tech 41:1227–1239
39. Rahimi-Vahed A, Mirzaei AH (2008) Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. Soft Comput 12:435–452
40. Rifai AP, Huu-Tho N, Dawal SZM (2016) Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl Soft Comput 40:42–57
41. Ruiz R, Vazquez-Rodriguez JA (2010) The hybrid flow shop scheduling problem. Eur J Oper Res 205:1–18
42. Sakawa M, Kubota R (2000) Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. Eur J Oper Res 120:393–407
43. Shao W, Shao Z, Pi D (2020) Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem. Knowl-Based Syst 2020:105527
44. Tang D, Dai M, Salido MA, Giret A (2016) Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. Comput Ind 81:82–95
45. Wang J, Wang L (2020) A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling. IEEE Trans Emerg Top Comput Intell 2020:1–15
46. Wang L, Fang C (2011) An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. Inf Sci 181:4804–4822
47. Wang L, Zhou G, Xu Y, Liu M (2013) A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. Int J Prod Res 51:3593–3608
48. Wang S, Liu M (2014) Two-stage hybrid flow shop scheduling with preventive maintenance using multi-objective tabu search method. Int J Prod Res 52:1495–1508
49. Xu Y, Wang L, Liu M, Wang, S.y., (2013) An effective shuffled frog-leaping algorithm for hybrid flow-shop scheduling with multiprocessor tasks. Int J Adv Manuf Tech 68:1529–1537
50. Yan J, Li L, Zhao F, Zhang F, Zhao Q (2016) A multi-level optimization approach for energy-efficient flexible flow shop scheduling. J Clean Prod 137:1543–1552
51. Ying KC, Lin SW (2018) Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. Expert Syst Appl 92:132–141
52. Zeng Z, Hong M, Man Y, Li J, Zhang Y, Liu H (2018) Multi-object optimization of flexible flow shop scheduling with batch process—consideration total electricity consumption and material wastage. J Clean Prod 183:925–939
53. Zhang X, Wang Y, Yan D, Ji Z (2017) Improved shuffled frog-leaping algorithm for solving flexible job shop scheduling problem. J Syst Simul 29:2093–2099
54. Zheng J, Wang L, Wang JJ (2020) A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. Knowl-Based Syst 2020:105536
55. Zitzler E, Thiele L (2000) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE T Evol Comput 3:257–271