Knowledge-Based Systems 194 (2020) 105536

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop☆

Jie Zheng, Ling Wang*, Jing-jing Wang

Department of Automation, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 17 October 2019
Received in revised form 16 December 2019
Accepted 16 January 2020
Available online 20 January 2020

Keywords:
Multi-objective fuzzy distributed hybrid flow shop
Fuzzy processing times and due dates
Robustness
Cooperative coevolution algorithm
Estimation of distribution algorithm
Iterated greedy search

ABSTRACT

With consideration of uncertainty in the distributed manufacturing systems, this paper addresses a multi-objective fuzzy distributed hybrid flow shop scheduling problem with fuzzy processing times and fuzzy due dates. To optimize the fuzzy total tardiness and robustness simultaneously, a cooperative coevolution algorithm with problem-specific strategies is proposed by reasonably combining the estimation of distribution algorithm (EDA) and the iterated greedy (IG) search. In the EDA-mode search, a problem-specific probability model is established to reduce the solution space and a sample mechanism is proposed to generate new individuals. To enhance exploitation, a specific local search is designed to improve performance of non-dominated solutions. Moreover, destruction and reconstruction methods in the IG-mode search are employed for further exploiting better solutions. To balance exploration and exploitation capabilities, a cooperation scheme for mode switching is designed based on the information entropy and the diversity of elite solutions. The effect of the key parameters on the performances of the proposed algorithm is investigated by Taguchi design of experiment method. Comparative results and statistical analysis demonstrate the effectiveness of the proposed algorithm in solving the problem.

1. Introduction

Under the trend of globalization, the multi-shop co-production pattern has become increasingly popular. Distributed manufacturing will conduce to share resources, enhance economic benefits and respond to market changes rapidly. In such a situation, a variety of distributed shop scheduling problems have been studied, such as the distributed flow shop scheduling [1–4], the distributed job shop scheduling [5] and the distributed flexible job shop scheduling [6]. Compared with the traditional single-shop problems, the distributed shop scheduling problems consider both assignment of jobs to different shops and job sequencing in each shop. Thus, it is more difficult to solve the distributed shop scheduling problems.

Nowadays, the hybrid flow shop scheduling problem (HFSP) has been widely applied in industries, such as electronic, glass, textiles, and semiconductors [7,8]. The HFSP is an extension of the classical permutation flow shop scheduling problem for the flexible manufacturing system where multiple parallel machines are employed at each stage. To solve the HFSP effectively, a lot of meta-heuristics have been proposed, including genetic algorithm (GA) [9], artificial bee colony algorithm (ABC) [10], iterated greedy (IG) algorithm [11], and estimation of distribution algorithm (EDA) [12]. To the best of our knowledge, few research works have been carried out on the distributed hybrid flow shop scheduling problem (DHFSP). To minimize the makespan of the DHFSP with multiprocessor tasks, Ying and Lin [13] proposed a mixed integer linear programming formulation and a self-tuning IG (SIG) algorithm by using an adaptive cocktail decoding mechanism. Hao et al. [14] presented a hybrid brain storm optimization algorithm with distributed Nawaz–Enscore–Ham (NEH) method [15] to solve the DHFSP.

In real world manufacturing systems, there are various uncertain factors, including model-inherent uncertainty such as kinetic constants, process-inherent uncertainty such as processing time, external uncertainty such as product demands, and discrete uncertainty such as machine maintenance [16]. Therefore, it is significant to study the DHFSP under uncertain environment. Fuzzy set theory is a common technique to handle uncertainty and vagueness [17]. During recent years, some research has been carried out on the scheduling problems with uncertain parameters by introducing the fuzzy set theory into the framework of the algorithms. For the fuzzy flexible job shop problem (FFJSP), Palacios et al. [18] proposed a cooperative coevolution algorithm (CCA), with the consideration of different ranking methods for

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2020.105536.

* Corresponding author.
E-mail address: wangling@mail.tsinghua.edu.cn (L. Wang).

https://doi.org/10.1016/j.knosys.2020.105536
0950-7051/

fuzzy numbers and the study of their influence on the solution robustness. Wang [19] presented an estimation of distribution algorithm, modeling job processing times as fuzzy numbers and employing fuzzy number operations to calculate the scheduling objective value. And Lin [20] combined biogeography-based optimization with some heuristics to solve the same problem. In addition, Gao et al. [21] considered the uncertainty of both processing times and new job insertion, and proposed a two-stage ABC based on rescheduling with fuzzy set theory. For HFSP with fuzzy processing times, Hong et al. [22] combined fuzzy largest processing time rule with fuzzy Johnson algorithm, but it cannot obtain satisfactory and robust solutions for large-scale problems. Rezaie et al. [23] presented a mixed-integer programming model for FHFSP to maximize the satisfaction level of meeting due dates, but it is not suitable for large-scale problems due to explosive growth of computing time. Behnamian and Ghomi [24] combined GA and particle swarm optimization to minimize both makespan and the sum of the earliness and tardiness for FHFSP. Zare and Fakhrzad [25] proposed a hybrid GA with the attribute-deductive data mining method. However, the robustness of the solution is not considered in the above research.

Taking into account both robustness and economic indicator, it becomes a multi-objective problem. The methods to solve the multi-objective problems can be roughly classified into four types: weighting approach, the constraint transformation method, Pareto-based approach, and the decomposition approach. For the stochastic HFSP, Wang et al. [26] assigned different weight for each objective to transfer the multi-objective problem to a single-objective one. For the aero-assisted vehicle trajectory planning problem, Chai et al. [27] embed the priority requirements into the model and transfer the original problem to a single-objective formulation. Deb et al. [28] proposed the non-dominated sorting genetic algorithm (NSGA-II), generating the Pareto front of a multi-objective problem by non-dominated sorting procedure and crowding distance metric. Chai et al. [29] applied the Pareto-based approach and proposed a game theory based adaptive differential evolution method with a control logic. Besides, Zhang et al. [30] presented a multi-objective evolutionary algorithm based on decomposition by decomposing a multi-objective optimization problem into a number of scalar sub-problems.

This paper takes first attempt to address the multi-objective FDHFSP (MFDHFSP) with fuzzy processing times and due dates. Since the MFDHFSP is a complex NP-hard multi-objective problem with uncertainty, large search space and strongly coupled sub-problems, we will propose a Pareto-based cooperative coevolution algorithm according to the characteristics of the problem. For uncertainty, we take into account the robustness of the solutions. To deal with multiple objectives, the Pareto-based approach is applied in the coevolution framework. To solve the strongly coupled sub-problems, i.e., shop allocation, job sequence and machine assignment, specific encoding and decoding methods with virtual jobs are utilized. Besides, the EDA-mode search and the IG-mode search are cooperated to balance exploration and exploitation of the algorithm. To reduce the search space, a problem-specific probability model and a suitable sampling mechanism are designed. Moreover, the local search based on critical shops is used to further enhance exploitation capability. The effect of the key parameters is investigated and the effectiveness of the proposed algorithm is demonstrated by extensive simulation comparisons.

The remainder of the paper is organized as follows. In Section 2, operations of fuzzy set and common concepts of Pareto-based approach are introduced, and the MFDHFSP are described. In Section 3, the CCA is presented in details. Simulation results and statistical analysis are provided in Section 4. Finally, we end the paper with some discussions, conclusions and future work in Section 5.

2. The fuzzy distributed hybrid flow shop problem

2.1. Operations on fuzzy processing time and fuzzy due date

To deal with fuzzy processing times and due dates in the MFDHFSP, the arithmetic operations of fuzzy numbers are used, including addition, subtraction, maximum, and ranking operation. For two triangular fuzzy numbers (TFNs) $A = (a^1, a^2, a^3)$ and $B = (b^1, b^2, b^3)$, addition, subtraction and maximum operations are shown as follows [31].

$$A + B = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \tag{1}$$

$$A - B = (a^1 - b^3, a^2 - b^2, a^3 - b^1) \tag{2}$$

$$\max(A, B) \approx \max_{I}(A, B) = \left(\max\left(a^1, b^1\right), \max\left(a^2, b^2\right), \max\left(a^3, b^3\right)\right) \tag{3}$$

where $\max_I$ is the approximation of $\max(A, B)$ [32].

The expected value of the TFN A is given as follows [33].

$$E(A) = (a^1 + 2a^2 + a^3)/4 \tag{4}$$

Besides, the following criterions are adopted to rank two TFNs [34].

Step1: The greatest number $Z_1(A) = E(A)$ will be chosen as the first criterion to rank two TFNs.

Step2: If there exist ties, $Z_2(A) = a^2$ is used as the second criterion.

Step3: If Z_1 and Z_2 are identical, $Z_3(A) = a^3 - a^1$ will be chosen as the third criterion.

2.2. Basic concepts of multi-objective optimization

Generally, a multi-objective optimization problem can be described as follows.

$$Minimize\ y = f(x) = (f_1(x), f_2(x), \ldots, f_{num}(x)) \tag{5}$$

where $x \in \Omega$ is the decision solution, Ω is the domain of definition, $f_i(x)(i = 1, \ldots, num)$ is the value of the ith objective, and y is the objective vector with num objectives.

In this paper, we employ the Pareto-based approach to deal with the multiple objectives and some common concepts are as follows.

Pareto dominance: A feasible solution a is said to dominate another feasible solution b if and only if $f_i(a) \le f_i(b), i = 1, 2, \ldots, num$, and $\exists i \in \{1, 2, \ldots, num\} : f_i(a) < f_i(b)$.

Pareto optimality: A feasible solution a is optimal if it is not dominated by any other feasible solutions.

Optimal Pareto set/Pareto Front: The solution set containing all optimal Pareto solutions is defined as the optimal Pareto set in solution space or the Pareto Front in objective space.

Non-dominated solution set/archive set (**AS**): The optimal Pareto set obtained by a certain algorithm is defined as a non-dominated solution set/archive set (AS).

2.3. Description of MFDHFSP

The MFDHFSP is descripted with following notations.

n: number of jobs
s: number of stages in one shop
F: number of shops
i: index of stages, $i = 1, \ldots, s$
j: index of jobs, $j = 1, \ldots, n$
m_i: number of machines at ith stage
l: index of machines, $l = 1, \ldots, m_i$
$M_{i,l}$: the lth machine at stage i
$v_{i,l}$: machine speed of $M_{i,l}$

$O_{i,j}$: the operator of job j at stage i
$p_{i,j}$: the fuzzy process time of job j at stage i, $p_{i,j} = (p^1_{i,j}, p^2_{i,j}, p^3_{i,j})$
d_j: the fuzzy due date of job j, $d_j = (d^1_j, d^2_j, d^3_j)$
$C_{i,j}$: the fuzzy completion time of $O_{i,j}$, $C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j})$
C_j: the fuzzy completion time of job j, $C_j = C_{s,j}$
T_j: the fuzzy tardiness of job j, $T_j = \max(C_j - d_j, zero)$
TT: the fuzzy total tardiness of a solution
ETT: the expected value of the fuzzy total tardiness
Rob^D: robustness indicator

The MFDHFSP comprises n jobs, F identical shops, and each shop contains s stages. Stage $i(i = 1, \ldots, s)$ contains m_i uniform parallel machines $\{M_{i,1}, M_{i,2}, \ldots, M_{i,mi}\}$ in each shop. The machine speeds $v_{i,l}(l = 1, \ldots, m_i)$ satisfy $1 \le v_{i,1} \le v_{i,2} \le \cdots \le v_{i,mi}, \forall i$.

Each job is assigned to one of the F shops, and sequentially processed at s stages, which is expressed by operations $O_{i,j}(i = 1, \ldots, s)$. The processing time of $O_{i,j}$ is represented as a TFN $p_{i,j} = (p^1_{i,j}, p^2_{i,j}, p^3_{i,j})$, where $p^1_{i,j}$ is the shortest processing time, $p^2_{i,j}$ is the most possible processing time and $p^3_{i,j}$ is the longest processing time. If $O_{i,j}$ is conducted on machine $M_{i,l}$, the actual processing time becomes $p_{i,j}/v_{i,l}$. Each job has a TFN due date $d_j = (d^1_j, d^2_j, d^3_j)$, where d^1_j is the earliest due date, d^2_j is the most possible due date and d^3_j is the latest due date. Similarly, the fuzzy completion time of $O_{i,j}$ is defined as a TFN $C_{i,j} = (C^1_{i,j}, C^2_{i,j}, C^3_{i,j})$, and the tardiness of job j equals to the TFN $T_j = \max(C_j - d_j, zero)$, where C_j is the fuzzy completion time of job j, $C_j = C_{s,j}$, and the fuzzy number $zero = (0, 0, 0)$.

In addition, it assumes that all jobs are independent, and their release times are 0; preemption is not allowed; every machine is continuously available and can only process one job at a time; every job is processed in the same sequence, and only can be processed on one machine at a time; and if a job is assigned to a shop, it cannot be reassigned to other shops afterwards. The objective functions include the minimum expected total tardiness (ETT) and the maximum robustness. The ETT is defined as the expected value of the TFN total tardiness $TT = (TT^1, TT^2, TT^3)$. The robustness denoted as Rob^D is defined as the maximum difference between the modal value TT^2 and the bounds of the fuzzy interval TT [33,34] as follows. Obviously, a smaller Rob^D implies better robustness.

$$TT = \left(TT^1, TT^2, TT^3\right) = \sum_{j=1}^{n} T_j \tag{6}$$

$$ETT = (TT^1 + 2TT^2 + TT^3)/4 \tag{7}$$

$$Rob^D = max(TT^2 - TT^1, TT^3 - TT^2) \tag{8}$$

3. The cooperative coevolution algorithm for the MFDHFSP

Cooperative coevolution is an effective way to improve the performance of evolutionary algorithms in solving complex optimization problems [35,36]. To solve the FJSP, Wang et al. [37] proposed a bi-population EDA by using population cooperation to adjust the machine assignment and operation sequence respectively. To solve the FFJSP, Palacios et al. [18] presented a cooperative coevolution algorithm with two sub-populations. As we known, it is important to balance global exploration and local exploitation for evolutionary algorithms. To solve the MFDHFSP effectively, we combine the probability model based EDA search and greedy strategy based IG search [38]. Besides, we also present a cooperation scheme for coevolution by reasonably fusing the EDA-mode search and the IG-mode search. Next, we will present the algorithm with encoding and decoding methods, population initialization, the EDA-mode search, the IG-mode search, the cooperation scheme and the complexity analysis.

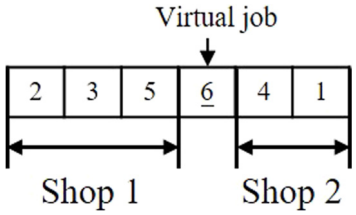

Fig. 1. Illustration of an encoded solution.

Table 1
Fuzzy processing times and machine speeds.

Job	Stage 1	Stage 2	Machine	Speed
1	[2, 3, 4]	[2, 3, 4]	$M_{1,1}$	1
2	[6, 10, 14]	[2, 3, 4]	$M_{1,2}$	2
3	[2, 4, 6]	[6, 10, 14]	$M_{2,1}$	1
4	[4, 8, 12]	[6, 8, 10]	$M_{2,2}$	2
5	[2, 5, 8]	[14, 16, 18]		

3.1. Encoding and decoding

In this paper, a solution is represented by $n+F-1$ permutation sequence Π, including $F-1$ virtual jobs and n actual jobs. Each virtual job is used to separate actual jobs to different shops and the sequences of actual jobs denote the processing orders at the first stage in each shop. For an example with 5 jobs and 2 shops, a feasible solution encoded as 2-3-5-6-4-1 denotes that jobs 2→3→5 are assigned to shop 1 and processed sequentially at the first stage while jobs 4→1 are assigned to shop 2. The virtual job 6 separates all the actual jobs into two shops. Such an encoded solution is illustrated in Fig. 1.

In each shop, jobs are assigned to the uniform machine with the earliest finish time. Besides, *List scheduling (LS) rule* [12] is applied to determine the job processing sequences after the first stage. According to *LS rule*, the jobs at stage i ($i = 2, \ldots, s$) are scheduled in an ascending order of their completion times at the previous stage.

For the above example, suppose the fuzzy processing times and speeds of all uniform parallel machines are given in Table 1. The Gantt chart of the schedule is shown in Fig. 2, where the triangles above the dotted lines are fuzzy start times of the corresponding operations, and the triangles below the dotted lines are their fuzzy completion times.

3.2. Population initialization

Considering both quality and diversity, a heuristic named modified NEH2 [15] (mNEH2) and a random method are used in a hybrid way to initialize the population. The procedure of the mNEH2 is as follows.

Step 1: Let initial sequence be $\Pi_0 = \{n+1, n+2, \ldots, n+F-1\}$;

Step 2: Rank n actual jobs according to their due dates in an ascending order, $J = (J_{(1)}, J_{(2)}, \ldots, J_{(n)})$. Let $j = 1$.

Step 3: Insert job $J_{(j)}$ to all possible positions in Π_0, and calculate the corresponding TT of the partial sequence. The partial sequence with lowest TT is denoted as Π_{new}. If there exist ties, then choose the one with minimum Rob^D as Π_{new}. If their Rob^D values are the same, then select the one with minimum fuzzy completion time as Π_{new}. Let $\Pi_0 = \Pi_{\text{new}}$.

Step 4: If $j<n, j = j + 1$, go to Step 3; otherwise, output Π_0.

By employing the mNEH2, a solution with certain quality can be generated. To ensure the diversity of initial population, other $Psize - 1$ solutions are generated randomly, where $Psize$ is the population size. Besides, the AS is initialized as the set of all non-dominated solutions in the initial population.

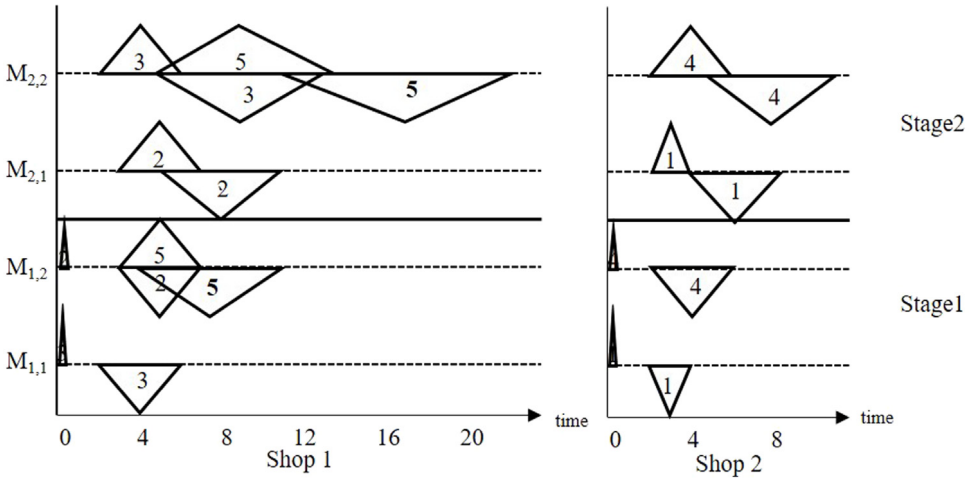

Fig. 2. Gantt chart of the encoded solution.

3.3. EDA-mode search

As a population-based evolutionary algorithm, the EDA [39] is of good global exploration capability. The main concept of the EDA is statistical learning, where a probability model is established to describe the distribution of solution space. Then a new population is generated by sampling the probability model, and the model is updated by the elite individuals of the population [12]. The EDA has been successfully used to solve resource allocation and scheduling problems in recent years [40,41]. In our algorithm, the EDA is adopted to explore promising solutions.

3.3.1. Probability model initialization

Since the MFDHFSP has a large solution space with the scale $O\left((n+F-1)! \times \prod_{i=1}^{s} m_i^n\right)$, it is important to employ knowledge -based strategy to reduce the search space. It has been proved that there must be at least one job in each shop in the optimal solution when $n > F$ [1]. By employing such a property, a problem-specific probability model and a sampling mechanism are designed, which are able to reduce the search space to $O(n! \times C_{n-1}^{n-F+1} \times \prod_{i=1}^{s} m_i^n)$.

The probability model Q of the EDA is designed to consider both the shop assignment and job sequence. Each element $q_{j,pos}(e)$ of Q represents the probability of job $J_j (j = 1, 2, \ldots, n+F-1)$ assigned to position pos in the sequence in the eth updating process. To ensure that every shop has at least one job, $Q(0) = \{q_{j,pos}(0)\}$ is initialized as follows.

$$q_{j,pos}(0) = \begin{cases} \frac{1}{n+F-1}, & j = 1, \ldots, n;\ pos = 1, \ldots, n+F-1 \\ \frac{1}{n-F+1}, & j = n+1, \ldots, n+F-1;\ pos = 2j-2n, \ldots, 2j-n-F \\ 0, & \text{else} \end{cases} \tag{9}$$

where the first n rows indicate that each actual job can be assigned to every position in the sequence. The last $F-1$ rows imply that each virtual job $J_j (j = n+1, n+2, \ldots, n+F-1)$ can only be assigned to the position from the $(2j-2n)$th ranging to the $(2j-n-F)$th in the sequence to ensure that every shop should have at least one job. It is obvious that the positions with probability 0 will not be sampled during the sampling process. Therefore, the probabilities of the same positions in the updated probability model will also be 0. That is, the probabilities of these positions are always 0 during the evolution process.

For each row of $Q(0)$, the probability of every possible position is set equally to ensure that the whole solution space can be sampled uniformly. Obviously, $\forall j, \sum_{pos=1}^{n+F-1} q_{j,pos}(0) = 1$.

3.3.2. Sampling mechanism

New individuals are generated by sampling the probability model $Q(e)$. The sampling process is described as follows.

First, set $Q' = Q(e)$ as a temporary probability matrix. Then, determine the position pos of job J_j in the permutation sequence by sampling Q' using the roulette wheel method. The elements in the posth column and jth row of Q' are set as 0 to ensure each position can only be occupied once. To guarantee that each shop has at least one job, the corresponding elements of other virtual jobs are set as 0 when a virtual job is assigned. Afterwards, normalize each row of Q'. Repeat the above steps until all the jobs are assigned. Thus, a new individual is generated. The pseudo code of the sampling process is given as Algorithm 1.

3.3.3. Updating probability model

In general, the probability model of the EDA is updated with the elite individuals. In this paper, the non-dominated sorting method based on crowding distance is used to select the elite individuals for the multi-objective problem. For all the non-dominated solutions of current population, their crowding distances are calculated as Algorithm 2.

Algorithm 2: Procedure of calculating crowding distances
Input non-dominated solutions NDS
Output crowding distances
  for each solution X in NDS
    $d(X)$=0; // $d(\ )$ denotes the crowding distance;
  end
  for each objective m //m=1,2 is the index of the objectives
    NDS '=sort NDS with mth objective in ascending order;
    // NDS '(i) is the ith solution in the current solution set;
    d(NDS '(1))=infinite;
    d(NDS '(N))=infinite; // N is the size of NDS;
    for i=2 **to** N–1
      $d(i)$=$d(i)$+NDS(i+1).m–NDS(i–1).m;
      // NDS (i).m denotes the value of mth objective of NDS (i).
    end
  end

The non-dominated solutions with largest $\eta\% \times Psize$ crowding distances constitute the elite set (ES), where $\eta\%$ is the ratio of elite solutions. If the number of the non-dominated solutions is less than $\eta\% \times Psize$, all of them constitute the ES. The elite solutions in the ES are used to update $Q(e)$ as follows.

$$q_{j,pos}(e) = (1-\alpha)\, q_{j,pos}(e-1) + \frac{\alpha}{\eta\% \times Psize} \sum_{h=1}^{\eta\% \times Psize} I_{jpos}^{h},\ \forall j, pos$$

Algorithm 1: Sampling Procedure
Input $Q(e)$
Output a permutation sequence Π
$Q'=Q(e)$; $j=n+F$-1;
while j>0 **do**
Sample $Q'(j,:)$ with roulette wheel method to determine position pos of J_j in the permutation sequence;
//$Q'(j,:)$ is the j-th row of Q'
$\Pi(pos)=j$;
for w=1 **to** $n+F$-1
$Q'(w,pos)$=0; $Q'(j,$w$)$=0;
end
$Q'(j,pos)$=1;
if $j>n$
$w=j$-1;
while $w>n$ **do**
for k=pos+2w-2j **to** 2w-n-F
$Q'(w,k)$ = 0;
end
$w=w$-1;
end
end
Normalize each row of Q';
$j=j$-1;
end

$$(10)$$

where $\alpha \in (0, 1)$ is the learning rate, and I^h_{jpos} is the following indicator function of hth individual in the ES.

$$I^h_{jpos} = \begin{cases} 1, & \text{if } J_j \text{ appears in position } pos \\ 0, & \text{else} \end{cases} \quad (11)$$

3.3.4. Local search based on critical shops

To improve the quality of the individuals in the ES, a local search is designed based on critical shops, including the shops with maximum and minimum TT denoted as $Smax$ and $Smin$, respectively. Besides, the job with maximum tardiness in $Smax$ is denoted as a critical job $Jmax$.

The local search performs the following three operators one by one on each individual X_i in the ES until no non-dominated solution is generated.

- Insert_inS: insert $Jmax$ into a random position in $Smax$ and generate a new solution X_i'. If X_i is dominated by X_i', X_i is replaced by X_i'.
- Swap_BS: randomly select a job in $Smax$ and another in $Smin$, then exchange their positions and generate a new solution X_i'. If X_i is dominated by X_i', X_i is replaced by X_i'.
- Insert_BS: insert $Jmax$ into a random position in $Smin$ and generate a new solution X_i'. If X_i is dominated by X_i', X_i is replaced by X_i'.

After the local search phase, the AS is updated by the newly non-dominated solutions.

3.4. IG-mode search

As a simple mate-heuristic, the IG algorithm [42] is of powerful exploitation capability. So far, the IG has been successfully applied to a variety of scheduling problems such as the flow shop scheduling [42], the HFSP [43], and the DHFSP [44]. In this paper, the IG is cooperated with the EDA to balance exploration and exploitation capabilities. The main procedure of the IG-mode search includes destruction and reconstruction.

The destruction phase consists of the following steps.

Step 1: Randomly select one actual job from the shops with largest α_{IG} total tardiness respectively, where α_{IG} is the number of removed jobs in the destruction phase. If $\alpha_{IG} > F$, randomly select the rest $\alpha_{IG} - F$ jobs from the remaining actual jobs.

Step 2: All the selected jobs are removed from the current permutation Π, and constitute the new sequence Π_S. The partial sequence of the remaining jobs (including virtual jobs) is denoted as Π_R.

The reconstruction phase is performed as follows.

Step 1: Let $j = 1$. Reinsert the jth job of Π_S into the best position of Π_R, and obtain the new partial sequence Π_R^*, where the best position is determined by non-dominated sorting.

Step 2: $\Pi_R = \Pi_R^*$. If $j<\alpha_{IG}$, $j = j + 1$, go to Step 1. Otherwise, output Π_R.

In the IG-mode search, the destruction and reconstruction methods are performed on each individual X_i in the AS. If the newly generated solution X_i' is not dominated by X_i, the AS will be updated by X_i'.

3.5. Cooperation scheme

To balance the EDA-based exploration and the IG-based exploitation, the cooperation scheme is designed based on the information entropy and the diversity of elite individuals in the AS.

3.5.1. Convergence evaluation of the EDA-mode search

Information entropy is used to measure the uncertainty of stochastic variables [44]. In this paper, we employ information entropy to evaluate the disorder degree of the probability model and the convergence of the EDA-mode search.

The information entropy of a random variable X is calculated as follows.

$$H(X) = \sum_i -p(x_i)\log_b(p(x_i)) \quad (12)$$

where x_i is the ith possible value of X, $p(x_i)$ is the probability of x_i, and $b = 2$ is the base of the logarithm.

For independent variables X_i $(i = 1, \ldots, n)$, their joint information entropy is calculated as follows.

$$H(X_1, X_2, \ldots, X_n) = -\sum_i \sum_j -p(x_{ij}) \log_b(p(x_{ij})) \tag{13}$$

where x_{ij} is the jth value of X_i and $p(x_{ij})$ is the probability of x_{ij}.

In the EDA-mode search, each element of the probability model $Q(e)$ can be viewed as a random variable. Accordingly, the information entropy of $Q(e)$ is expressed as Eq. (14).

$$H(Q(e)) = -\sum_{j=1}^{n+F-1} \sum_{pos=1}^{n+F-1} q_{j,pos}(e) \log 2 \left(q_{j,pos}(e)\right) \tag{14}$$

Thus, the information entropy of $Q(0)$ can be derived as follows, denoted as H_0.

$$\begin{aligned} H_0 = & -\sum_{j=1}^{n} \sum_{pos=1}^{n+F-1} \frac{1}{n+F-1} \log 2 \left(\frac{1}{n+F-1}\right) \\ & - \sum_{j=n+1}^{n+F-1} \sum_{pos=2j-2n}^{2j-n-F} \frac{1}{n-F+1} \log 2 \left(\frac{1}{n-F+1}\right) \\ = & n \log_2 (n+F-1) + (F-1) \log 2(n-F+1) \end{aligned} \tag{15}$$

Since $H(Q(e))$ can measure the uncertainty of $Q(e)$, $H(Q(e))$ will be close to a certain number when $Q(e)$ converges. Therefore, if the information entropy of two adjacent generations varies slightly, the probability model is basically unchanged which can be regarded as the convergence of the EDA-mode search to some extent. So, we design a terminal condition 1 (TC1) according to the change of information entropy as follows. If TC1 holds, the algorithm will switch to IG-mode search.

$$\frac{|H_i - H_{i-1}|}{H_0} \leq 0.01 \text{ and } H_i < H_{i-1} \tag{16}$$

where H_i is the information entropy of $Q(e)$ in the ith iteration of the EDA-mode search.

3.5.2. Diversity evaluation of the IG-mode search

In the IG-mode search, the diversity of the individuals in the AS can be evaluated by the number of solutions in the AS as well as the number of newly generated non-dominated solutions. So, we design a terminal condition 2 (TC2) for the IG-mode search as follows. If the number of solutions in the AS is less than $\eta\% \times Psize$ or there is no new non-dominated solution generated by the IG-mode search, then the algorithm will switch to the EDA-mode search. Once the mode is switched, $Q(e)$ will be updated by the elite solutions in the AS, i.e., the best $\eta\% \times Psize$ solutions.

With the above switch scheme, the algorithm can adaptively choose to perform the EDA-mode search or the IG-mode search. Thus, it can avoid the slow convergence of the pure EDA-mode search and the premature convergence (loss of diversity) of the pure IG-mode search. Therefore, the global exploration and local exploitation can be well balanced. The mechanism of the cooperative coevolution algorithm is illustrated in Fig. 3 and the procedure of the CCA is shown in Algorithm 3.

3.6. Complexity analysis

Suppose that there are n jobs, s stages and F shops in the MFDHFSP. The computation complexity of computing TT is $O(ns)$. In the initial phase, the computational complexity of the mNEH2 and random method are $O(n \log n + ns \times n^2)$ and $O(ns \times n \times (Psize - 1))$ respectively. Therefore, the computation complexity of the initial phase is approximately equal to $O(n^3 s + n^2 s \times Psize)$.

In each iteration of the loop in the EDA-mode search, it needs to sample model, select elite solutions, local search, update model and calculate information entropy. According to Algorithm 1, the computation complexity of sampling is $O(Psize \times ns \times (n \times (n + F - 1) + (F - 1)^2 \times (n - F))) \approx O(Psize \times (n^3 s + n_2 s F^2 - snF^3))$. Selecting Elite solutions consists of 3 steps: find non-dominated solutions, calculate the crowding distance and select the first $\eta\% \times Psize$ solutions with computation complexity $O(Psize^2)$, $O(En \times \log En + En)$ and $O(En \times \log(\eta\% \times Psize))$, respectively, where $En < Psize$ is the number of non-dominated solutions. Besides, the computation complexities of updating model and calculating information entropy can be derived as $O(\eta\% \times Psize \times (n+F-1)+(n+F-1)^2)$ and $O((n+F-1)^2)$ respectively. As for local search, the computation complexity of one iteration is $O(Psize \times ns)$. Therefore, the computation complexity of the EDA-mode search in one iteration is as follows, where ls is the run times of the local search.

$$\begin{aligned} O(n, s, F, Psize, \eta, En, ls) \approx & O\left(Psize \times \left(n^3 s + n^2 s F^2 - snF^3\right)\right) \\ & + O\left(Psize^2\right) + O(En \times \log En + En) \\ & + O(En \times \log(\eta\% \times Psize)) + O(\eta\% \times Psize \times (n + F - 1) \\ & + (n + F - 1)^2) + O\left((n + F - 1)^2\right) + O(ls) \times O(Psize \times ns) \\ \approx & O\left(Psize \times ns\left(n^2 + nF^2 - F^3 + ls\right) + Psize^2\right) \end{aligned}$$

In each iteration of the loop in the IG-mode search, the algorithm needs to perform destruction and construction. The computation complexity of sorting and job removing in destruction are $O(n\log(\alpha_{IG}))$ and $O(ns)$. The computation complexity of construction for one solution is $O(ns \times \sum_{i=1}^{\alpha_{IG}} (n - \alpha_{IG} + i)^2)$. Because $\alpha_{IG} \leq n$, the computation complexity can be simplified as $O(\alpha_{IG} \times n^3 s)$. Therefore, the computation complexity of the IG-mode search in one iteration is $O(b \times \alpha_{IG} \times n^3 s)$, where b is the size of the AS.

In summary, the computation complexity of the CCA with k_1 iterations of EDA-mode search and k_2 iterations of IG-mode search is shown as follows.

$$\begin{aligned} O(k_1, k_2, \alpha_{IG}, n, F, s, ls) = & O\left(n^3 s + n^2 s \times Psize\right) \\ & + O(k_1) O\left(Psize \times ns\left(n^2 + nF^2 - F^3 + ns \times ls\right) + Psize^2\right) \\ & + O(k_2) O\left(b \times \alpha_{IG} \times n^3 s\right) \end{aligned}$$

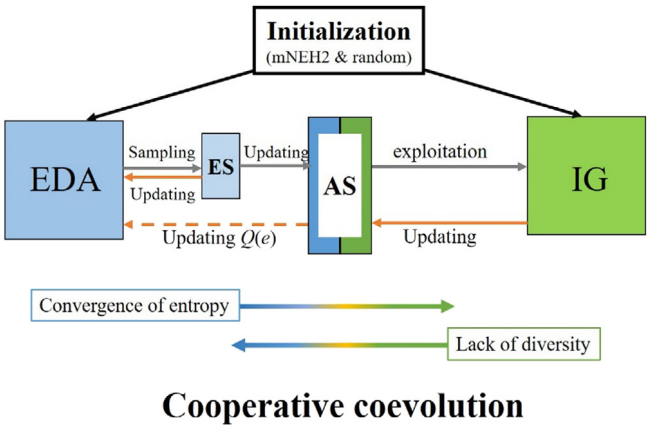

Fig. 3. The mechanism of the CCA.

4. Experimental results and comparisons

In this section, extensive numerical tests are conducted to evaluate the performances of the proposed algorithm. The testing instances are generated by extending the instances from [13] for the DHFSP. To be specific, the speeds of machines at each stage are set as a random number uniformly distributed in [1, 1.5] denoted as $v_{i,l} \sim U(1, 1.5)$, satisfying $1 \leq v_{i,1} \leq v_{i,2} \leq \cdots \leq v_{i,mi}, \forall i$.

Algorithm 3: Procedure of the CCA
Input A MFDHFSP instance
Output the archive set (AS)
Initialize the population by mNEH2 and random method;
Initialize the probability model $Q(0)$;
Calculate the initial entropy H_0;
Find non-dominated solutions of the population and update AS; Let e=0;
while terminal condition is not satisfied **do**
  i=1;
  while TC1 is not satisfied **do** // the EDA-mode search
    Generate *Psize* individuals by sampling $Q(e)$;
    Find non-dominated solutions of the new generated individuals;
    Calculate the crowding distance and select elite solutions to form the ES;
    for each individual in the ES
      Perform local search based on critical shops;
    end
    e=e+1;
    Update $Q(e)$ by the ES and update the AS;
    Calculate information entropy H_i of $Q(e)$;
    i=i+1;
  end
  while TC2 is not satisfied **do** // the IG-mode search
    for each individual in the *AS*
      Perform destruction and reconstruction;
    end
    Update the AS;
  end
  e=e+1; Update $Q(e)$ by elite solutions in the AS
end
return AS

The processing times of jobs are described as symmetric TFNs according to literature [32]. That is, $p_{i,j} = (p^1_{i,j}, p^2_{i,j}, p^3_{i,j})$, $p^2_{i,j} = p^1_{i,j} + p$, $p^3_{i,j} = p^1_{i,j} + 2p$, where $p \sim U(1, 5)$ is a random number and $p^1_{i,j}$ is the original deterministic processing time. Besides, the due dates are set as $d_j = [(1+\lambda \times \mu) \times \sum_s^i p_{ij}]$, where $\mu \sim U(0, 1)$, and $\lambda \in \{0.5, 2, 4\}$ is the tardiness factor, representing the tight, middle and loose cases respectively [45]. The algorithm is coded in C++ and run on a PC with Intel(R) Core(TM) i7-8700 CPU @ 3.2 GHz.

For the MFDHFSP, the performances of the solution algorithms should be evaluated in terms of the quantity, quality and distribution of the obtained archive set. In this paper, three widely used metrics [46] are employed.

ONVG: the metric overall non-dominated vector generation (ONVG) is defined as the number of distinct non-dominated solutions in set A, denoted as $|A|$. Intuitively, it reflects the quantity of the non-dominated solution set. A larger ONVG means there are more choices for the decision maker.

CM: the C metric (CM) is used to reflect the dominance relationship between two non-dominated sets A_1 and A_2 by comparing the quality of the solutions in the two sets.

$$C(A_1, A_2) = \frac{|\{x_2 \in A_2 | \exists x_1 \in A_1, x_2 \prec x_1 \text{ or } x_2 = x_1\}|}{|A_1|} \tag{17}$$

where $C(A_1, A_2)$ denotes the percentage of the solutions in A_2 that are dominated by or the same as the solution in A_1. The closer $C(A_1, A_2)$ is to 1, the more approximate A_1 is to the optimal Pareto set.

TS: the TS calculated is used to measure how evenly the solutions are distributed.

$$TS = \sqrt{\frac{1}{|A|} \sum_i^{|A|} \frac{(D_i - \overline{D})^2}{\overline{D}}} \tag{18}$$

where $\overline{D} = \sum_{i=1}^{|A|} D_i / |A|$, and D_i is the Euclid distance in objective space between the solution i and its nearest neighbor solution. The smaller the TS is, the more uniformly the solutions are distributed.

4.1. Parameters setting

The CCA contains four key parameters: the population size *Psize*, the learning rate of $Q(e)$, i.e. α, the radio of elite solutions η%, and the number of selected jobs in IG-mode search, i.e. α_{IG}. To investigate the effect of these parameters on the performance of the algorithm, we implement the Taguchi method of design of experiment [47] by using a moderate scale instance n50_s5_F3_1 ($n = 50, s = 5, F = 3, \lambda = 0.5$).

Four levels of each parameter are set as shown in Table 2. According to the number of parameters and levels, an orthogonal table $L_{16}(4^4)$ is generated. Thus, there are 16 combinations of parameter values in total. For each combination, we run the CCA 10 times independently on n50_s5_F3_1 and set $0.3 \times n \times s$ seconds as the termination condition. Then it obtains a composite non-dominated solution set for each combination. Furthermore, the final non-dominated solution set is obtained by aggregating the 16 composite sets of all combinations. Subsequently, the proportion of the non-dominated solution set obtained by a composite set of a combination is calculated as the response value (RV). Obviously, the larger the RV, the better the combination of parameter values. The orthogonal table and response values are listed in Table 3. The significant rank of each parameter is listed in Table 4. And the trend of each factor level is shown in Fig. 4.

From Table 4, it can be seen that *Psize* is the most significant parameter. If *Psize* is too small, the solution space cannot be sampled sufficiently. However, the algorithm will converge slowly if *Psize* is too large. Besides, α_{IG} ranks the second. A proper value of α_{IG} ensures sufficient local search in the IG-mode search. The learning rate α ranks the third. A large value of α could lead to premature convergence while a small value of α could

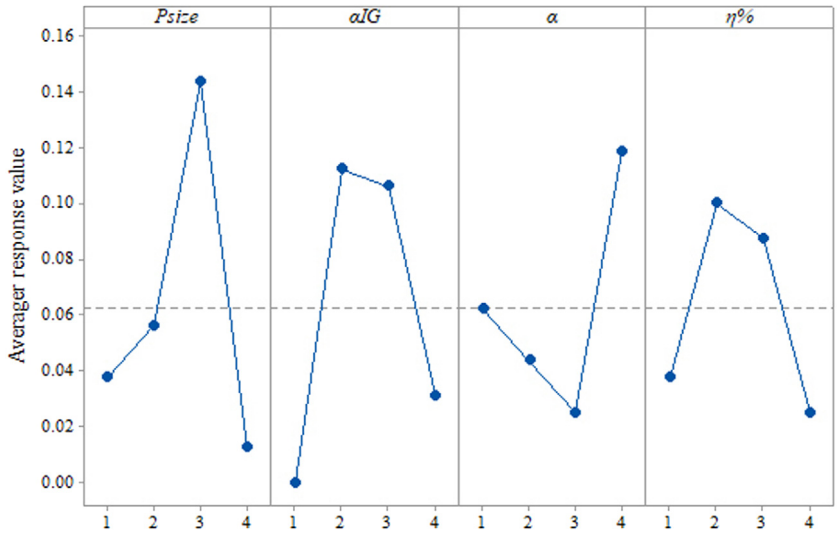

Fig. 4. Factor level trend of the CCA.

Table 2
Combinations of parameter values.

Level	*Psize*	α_{IG}	α	η%
1	20	1	0.2	0.2
2	30	2	0.3	0.3
3	40	3	0.4	0.4
4	50	4	0.5	0.5

Table 3
Orthogonal array and RV values.

No.	*Psize*	α_{IG}	α	η%	RV
1	1	1	1	1	0.000
2	1	2	2	2	0.100
3	1	3	3	3	0.025
4	1	4	4	4	0.025
5	2	1	2	3	0.000
6	2	2	1	4	0.025
7	2	3	4	1	0.125
8	2	4	3	2	0.075
9	3	1	3	4	0.000
10	3	2	4	3	0.325
11	3	3	1	2	0.225
12	3	4	2	1	0.025
13	4	1	4	2	0.000
14	4	2	3	1	0.000
15	4	3	2	4	0.050
16	4	4	1	3	0.000

Table 4
Response value and rank of each parameter.

Level	*Psize*	α_{IG}	α	η%
1	0.038	0.000	0.063	0.038
2	0.056	0.113	0.044	0.100
3	0.144	0.106	0.025	0.088
4	0.013	0.031	0.119	0.025
Delta	0.131	0.113	0.094	0.075
Rank	1	2	3	4

lead to slow convergence. As for η%, although it has the slightest influence, an appropriate value can help the algorithm update the probability model reasonably. According to above analysis, the parameters of the CCA are suggested as $Psize = 40, \alpha_{IG} = 2, \alpha = 0.5$, and $\eta = 0.3$, which will be used in the following experiments.

4.2. Simulation comparisons

In this section, we consider the problems with n jobs, s stages, and F shops, where $n \in \{20, 50, 100\}, s \in \{2, 5, 8\}$ and $F \in \{2, 3, 4, 5, 6\}$. For each combination of n, s, and F, 10 instances are generated, resulting $3 \times 3 \times 5 \times 10 = 450$ instances. Moreover, we set 3 levels of due dates for every instance. Thus, 1350 instances in total will be tested.

To the best of our knowledge, there is no published work addressing the MFDHFSP. Therefore, we generalize EDA [2] for DFSP, SIG [13] for DHFSP and NSGA-II [28] [48] to solve the MFDHFSP. To compare the CCA and the above three algorithms, we run each algorithm 5 times independently for each instance with $0.3 \times n \times s$ seconds CPU time as terminal condition. The results are listed in Tables 5–7 grouped by different tardiness factor λ, where C, E, S, N represent the CCA, EDA, SIG, NSGA-II respectively. Moreover, the nonparametric tests (Kruskal–Wallis tests) [49] with 95% confidence interval are carried out for each pair of *CM*. If p-value is smaller than 0.05, it implies the difference between the compared *CM* is significant.

From Tables 5–7, it can be seen that the CCA is better than other three algorithms in terms of *CM* on most groups of instances. As n increases, the superiority of the CCA becomes more obvious. For the groups with same n, the differences between C(C, E) and C(E, C) as well as C(C, N) and C(N, C) are larger with smaller value of F. So, it can be concluded that the CCA performs significantly better than the other three algorithms, especially on the instances with large n and small F. In addition, the p-values show the superiority of the CCA when $\lambda = 0.5$. However, when $\lambda = 2$ or 4 the p-values on some instances are larger than 0.05 or "−", where "−" means the objective values obtained by the two tested algorithms are the same. It implies that these instances can be solved relatively easily at loose due dates.

For ONVG and TS metrics, the comparative results are given in Table 8. It can be seen that ONVG of the EDA is better than that of SIG especially when $\lambda = 0.5$ while the results are opposite on instances with large value of F when $\lambda = 2$ or 4. The main reason is that the EDA has a better ability of exploration than the SIG. However, most solutions generated by the EDA can be dominated with loose due dates when $\lambda = 2$ or 4. Since the CCA combines the advantages of the EDA-mode search and the IG-mode search, the results of the CCA is the best on all groups of instances, which means it can find more non-dominated solutions. As for TS, from

Table 5
Results of the CM ($\lambda = 0.5$).

$n \times F$	CCA vs EDA			CCA vs SIG			CCA vs NSGA-II		
	C(C, E)	C(E, C)	p-value	C(C, S)	C(S, C)	p-value	C(C, N)	C(N, C)	p-value
20 × 2	**0.99**	0.04	**0.00**	**0.88**	0.11	**0.00**	**0.88**	0.04	**0.00**
20 × 3	**0.94**	0.19	**0.00**	**0.79**	0.31	**0.00**	**0.79**	0.19	**0.00**
20 × 4	**0.92**	0.33	**0.00**	**0.83**	0.36	**0.00**	**0.83**	0.27	**0.00**
20 × 5	**0.97**	0.44	**0.00**	**0.89**	0.50	**0.00**	**0.89**	0.48	**0.00**
20 × 6	**0.96**	0.56	**0.00**	**0.91**	0.58	**0.01**	**0.91**	0.58	**0.01**
50 × 2	**0.99**	0.01	**0.00**	**0.99**	0.00	**0.00**	**0.99**	0.00	**0.00**
50 × 3	**1.00**	0.00	**0.00**	**0.93**	0.05	**0.00**	**0.93**	0.02	**0.00**
50 × 4	**0.99**	0.00	**0.00**	**0.91**	0.06	**0.00**	**0.91**	0.00	**0.00**
50 × 5	**0.99**	0.03	**0.00**	**0.86**	0.13	**0.00**	**0.86**	0.00	**0.00**
50 × 6	**0.99**	0.07	**0.00**	**0.91**	0.15	**0.00**	**0.91**	0.03	**0.00**
100 × 2	**0.96**	0.02	**0.00**	**0.99**	0.00	**0.00**	**0.99**	0.03	**0.00**
100 × 3	**0.86**	0.04	**0.00**	**0.86**	0.07	**0.00**	**0.86**	0.06	**0.00**
100 × 4	**0.95**	0.01	**0.00**	**0.84**	0.06	**0.00**	**0.84**	0.01	**0.00**
100 × 5	**0.87**	0.06	**0.00**	**0.48**	0.30	0.06	**0.48**	0.07	**0.00**
100 × 6	**0.94**	0.01	**0.00**	**0.67**	0.17	**0.00**	**0.67**	0.02	**0.00**

Table 6
Results of the CM ($\lambda = 2$).

$n \times F$	CCA vs EDA			CCA vs SIG			CCA vs NSGA-II		
	C(C,E)	C(E,C)	p-value	C(C,S)	C(S,C)	p-value	C(C,N)	C(N,C)	p-value
20 × 2	**0.99**	0.68	**0.00**	**0.94**	0.72	0.11	**0.94**	0.64	**0.03**
20 × 3	**0.96**	0.77	0.06	**0.91**	0.84	0.78	**0.91**	0.77	0.47
20 × 4	**0.98**	0.83	0.10	**0.93**	0.86	0.66	**0.93**	0.83	0.78
20 × 5	**1.00**	**1.00**	–	**1.00**	0.98	0.33	**1.00**	**1.00**	–
20 × 6	**1.00**	**1.00**	–	**1.00**	**1.00**	–	**1.00**	0.97	0.33
50 × 2	**1.00**	0.13	**0.00**	**0.94**	0.19	**0.00**	**0.94**	0.07	**0.00**
50 × 3	**0.99**	0.17	**0.00**	**0.82**	0.25	**0.00**	**0.82**	0.13	**0.00**
50 × 4	**1.00**	0.27	**0.00**	**0.89**	0.47	**0.01**	**0.89**	0.23	**0.00**
50 × 5	**0.96**	0.45	**0.00**	0.68	0.77	0.46	**0.68**	0.38	**0.02**
50 × 6	**0.93**	0.68	**0.01**	**0.88**	0.79	0.45	**0.88**	0.58	**0.01**
100 × 2	**0.87**	0.06	**0.00**	**0.92**	0.03	**0.00**	**0.92**	0.07	**0.00**
100 × 3	**0.91**	0.05	**0.00**	**0.73**	0.16	**0.00**	**0.73**	0.11	**0.00**
100 × 4	**0.98**	0.03	**0.00**	**0.64**	0.23	**0.00**	**0.64**	0.01	**0.00**
100 × 5	**0.86**	0.20	**0.00**	**0.46**	0.45	0.84	**0.46**	0.15	**0.00**
100 × 6	**0.90**	0.23	**0.00**	**0.60**	0.44	0.17	**0.60**	0.15	**0.00**

Table 7
Results of the CM ($\lambda = 4$).

$n \times F$	CCA vs EDA			CCA vs SIG			CCA vs NSGA-II		
	C(C,E)	C(E,C)	p-value	C(C,S)	C(C,E)	p-value	C(E,C)	C(C,S)	p-value
20 × 2	**1.00**	0.97	0.33	**1.00**	0.99	1.00	**0.98**	**0.98**	1.00
20 × 3	**0.99**	0.93	0.54	**0.99**	0.97	1.00	**0.99**	0.91	0.29
20 × 4	**1.00**	0.97	0.33	**1.00**	0.97	0.33	**1.00**	0.97	0.33
20 × 5	**1.00**	**1.00**	–	**1.00**	**1.00**	–	**1.00**	**1.00**	–
20 × 6	**1.00**	**1.00**	–	**1.00**	**1.00**	–	**1.00**	**1.00**	–
50 × 2	**1.00**	0.37	**0.00**	**0.77**	0.62	0.23	**0.77**	0.43	**0.01**
50 × 3	**1.00**	0.53	**0.00**	**0.76**	0.75	0.84	**0.76**	0.47	**0.04**
50 × 4	**0.97**	0.78	**0.03**	**0.93**	0.83	0.43	**0.93**	0.65	**0.02**
50 × 5	**0.97**	0.92	0.32	**0.94**	**0.94**	1.00	**0.94**	0.88	0.65
50 × 6	**0.98**	0.94	0.96	**0.99**	0.93	0.96	**0.99**	0.90	0.59
100 × 2	**0.87**	0.06	**0.00**	**0.84**	0.12	**0.00**	**0.84**	0.06	**0.00**
100 × 3	**0.94**	0.20	**0.00**	**0.64**	0.41	**0.03**	**0.64**	0.19	**0.00**
100 × 4	**0.87**	0.31	**0.00**	**0.65**	0.53	0.67	**0.65**	0.18	**0.00**
100 × 5	**0.93**	0.48	**0.00**	0.60	**0.78**	0.09	**0.60**	0.38	**0.05**
100 × 6	**0.95**	0.61	**0.01**	**0.87**	0.78	0.30	**0.87**	0.55	**0.00**

Table 8 it can be seen that the results of the CCA are better than both EDA and NSGA-II on most instances and better than SIG on average. That is, the non-dominated solutions obtained by the CCA are of better distribution.

In addition, the Pareto fronts obtained by the four algorithms when solving the instance n100_s8_F3_1 ($n = 100, s = 8, F = 3, \lambda = 0.5$) are illustrated in Fig. 5. Clearly, the Pareto front obtained by the CCA is the best in terms of quantity, quality and distribution.

5. Discussions and conclusions

In this paper, we address the multi-objective fuzzy distributed hybrid flow shop problem with the minimization of the total tardiness and the maximization of the robustness. A coevolution algorithm composed of EDA and IG with a reasonable cooperation scheme is proposed to solve this problem. From extensive numerical tests, it can be seen that the convergence of our algorithm is significantly better than other algorithms on the instances with tight due dates. If the due dates are relatively loose, the diversity

Table 8
Results of the algorithms of ONVG and TS.

λ	F	ONGA				TS			
		CCA	EDA	SIG	NSGA-II	CCA	EDA	SIG	NSGA-II
0.5	2	**26.52**	18.56	13.22	26.39	10.31	17.61	10.06	**9.28**
	3	**24.90**	16.50	14.74	22.58	7.71	15.02	**7.67**	9.13
	4	**23.93**	15.47	14.16	19.16	**7.66**	11.39	8.10	7.99
	5	**21.06**	13.31	14.22	16.92	**4.94**	8.90	5.50	5.62
	6	**20.20**	12.98	12.11	16.22	**4.05**	7.70	4.33	5.89
2	2	**20.67**	13.78	9.90	20.36	**8.29**	15.66	8.54	9.14
	3	**15.94**	11.33	10.80	15.42	**5.28**	12.48	5.75	6.58
	4	**13.69**	8.18	9.38	11.30	4.21	7.65	**3.91**	5.33
	5	**11.48**	6.69	6.78	7.81	**2.45**	5.51	2.49	2.90
	6	**9.73**	5.11	6.27	6.77	**2.33**	4.35	2.86	2.90
4	2	**14.72**	9.16	7.93	14.24	**5.55**	16.08	6.08	6.43
	3	**9.24**	5.79	6.52	7.88	**3.64**	9.80	3.67	3.75
	4	**7.18**	4.01	4.86	5.61	1.93	4.52	**1.46**	2.08
	5	**3.77**	2.47	3.09	3.60	1.77	2.26	**0.86**	1.26
	6	**3.58**	2.07	2.56	2.53	**0.62**	1.76	0.70	0.66
Average		**15.11**	9.69	9.10	13.12	**4.72**	9.38	4.80	5.26

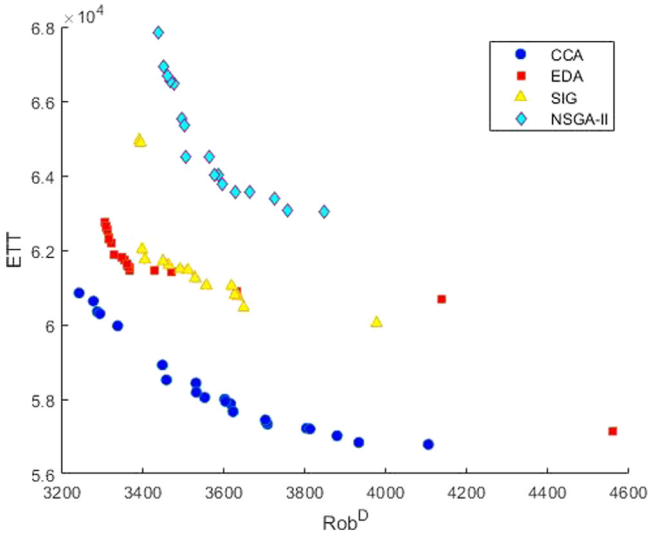

Fig. 5. Pareto fronts obtained by the four algorithms.

and distribution of the solutions obtained by our algorithm are obviously better. So, our algorithm is able to obtain feasible solutions with good quality, quantity and distribution for the above case. The superiority of our algorithm mainly owes to the follows.

(1) Cooperation of the EDA-mode search and the IG-mode search to balance exploration and exploitation of the algorithm.

(2) Utilization of the specific encoding and decoding methods with virtual jobs to deal with the strong coupling of shop allocation, job sequence and machine assignment.

(3) Hybridization of the heuristic and random method to produce a population with good quality and diversity.

(4) Utilization of the problem-specific probability model and sampling mechanism to effectively reduce the search space.

(5) Utilization of local search based on critical shops to enhance exploitation capability.

The above problem-specific designs make our algorithm suitable for such a complex scheduling problem. Since our algorithm is a population-based coevolution algorithm, some random operators are used to maintain the diversity of the population. Potential strategy to remove the negative effect of the randomness would enhance the practicability. Meanwhile, it will benefit from parameter variations. So, in our future work we will study adaptive strategies for parameter control and operator utilization. Especially, we will focus on the knowledge fusion optimization algorithms by studying knowledge utilization and learning mechanism to further enhance the performance. In addition, it is interesting to solve the distributed scheduling problems with other types of uncertainties, and it is also important to generalize the problems with more objectives including other robustness criteria.

CRediT authorship contribution statement

Jie Zheng: Conceptualization, Data curation, Formal analysis, Methodology, Writing - original draft, Writing - review & editing. **Ling Wang:** Conceptualization, Data curation, Formal analysis, Methodology, Writing - original draft, Writing - review & editing. **Jing-jing Wang:** Conceptualization, Data curation, Formal analysis, Methodology, Writing - original draft, Writing - review & editing.

Acknowledgments

This research is supported by the National Science Fund for Distinguished Young Scholars of China [No. 61525304], the National Natural Science Foundation of China [No. 61873328], and Meituan-Dianping Group.

References

[1] B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, Comput. Oper. Res. 37 (4) (2010) 754–768.
[2] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem, Int. J. Prod. Econ. 145 (1) (2013) 387–396.
[3] W. Shao, D. Pi, Z. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms, Knowl.-Based Syst. 137 (2017) 163–181.
[4] T. Meng, Q.K. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, Knowl.-Based Syst. 184 (2019) 104894.
[5] C.Y. Hsu, B.R. Kao, V.L. Ho, K.R. Lai, Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling, Eng. Appl. Artif. Intell. 53 (2016) 140–154.
[6] L.D. Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, European J. Oper. Res. 200 (2) (2010) 395–408.
[7] R. Ruiz, J.A. Vázquez-Rodríguez, The hybrid flow shop scheduling problem, European J. Oper. Res. 205 (1) (2010) 1–18.
[8] I. Ribas, R. Leisten, J.M. Framiñan, Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective, Comput. Oper. Res. 37 (8) (2010) 1439–1454.
[9] O. Engin, G. Ceran, M.K. Yilmaz, An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems, Appl. Soft Comput. 11 (3) (2011) 3056–3065.
[10] Q.K. Pan, L. Wang, J.Q. Li, J.H. Duan, A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation, Omega 45 (2014) 42–56.
[11] K.C. Ying, An iterated greedy heuristic for multistage hybrid flowshop scheduling problems with multiprocessor tasks, J. Oper. Res. Soc. 60 (6) (2009) 810–817.
[12] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines, Int. J. Adv. Manuf. Technol. 68 (9–12) (2013) 2043–2056.
[13] K.C. Ying, S.W. Lin, Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks, Expert Syst. Appl. 92 (2018) 132–141.
[14] J.H. Hao, J.Q. Li, Y. Du, M.X. Song, P. Duan, Y.Y. Zhang, Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm, IEEE Access (2019) http://dx.doi.org/10.1109/ACCESS.2019.2917273.
[15] M. Nawaz, E.E.J. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow shop sequencing problem, Omega 11 (1) (1983) 91–95.
[16] E. Pistikopoulos, Uncertainty in process design and operations, Comput. Chem. Eng. 19 (1995) 553–563.
[17] P. Fortemps, Fuzzy sets in scheduling and planning, European J. Oper. Res. 147 (2) (2003) 229–230.

[18] J.J. Palacios, I. González-Rodríguez, C.R. Vela, J. Puente, Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop, Fuzzy Sets and Systems 278 (2015) 81–97.
[19] S. Wang, L. Wang, Y. Xu, M. Liu, An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time, Int. J. Prod. Res. 51 (12) (2013) 3778–3793.
[20] J. Lin, A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem, Knowl.-Based Syst. 78 (2015) 59–74.
[21] K.Z. Gao, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, A. Sadollah, Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion, Knowl.-Based Syst. 109 (2016) 1–16.
[22] T.P. Hong, T.T. Wang, Fuzzy flexible flow shops at two machine centers for continuous fuzzy domains, Inform. Sci. 129 (1–4) (2000) 227–237.
[23] N. Rezaie, R. Tavakkoli-Moghaddam, S. Torabi, A new mathematical model for fuzzy flexible flow shop scheduling of unrelated parallel machines maximizing the weighted satisfaction level, IFAC Proc. Vol. (IFAC PapersOnline) 42 (4) (2009) 798–803.
[24] J. Behnamian, S.F. Ghomi, Multi-objective fuzzy multiprocessor flowshop scheduling, Appl. Soft Comput. 21 (2014) 139–148.
[25] H.K. Zare, M.B. Fakhrzad, Solving flexible flow-shop problem with a hybrid genetic algorithm and data mining: A fuzzy approach, Expert Syst. Appl. 38 (6) (2011) 7609–7615.
[26] S.Y. Wang, L. Wang, M. Liu, Y. Xu, An order-based estimation of distribution algorithm for stochastic hybrid flow-shop scheduling problem, Int. J. Comput. Integr. Manuf. 28 (3) (2014) 307–320.
[27] R.Q. Chai, A. Savvaris, A. Tsourdos, S. Chai, Y. Xia, Unified multiobjective optimization scheme for aeroassisted vehicle trajectory planning, J. Guid. Control Dyn. 41 (7) (2018) 1521–1530.
[28] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[29] R.Q. Chai, A. Savvaris, A. Tsourdos, S. Chai, Multi-objective trajectory optimization of space manoeuvre vehicle using adaptive differential evolution and modified game theory, Acta Astronaut. 136 (2017) 273–280.
[30] Q.F. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[31] S. Chanas, A. Kasperski, On two single machine scheduling problems with fuzzy processing times and fuzzy due dates, European J. Oper. Res. 147 (2) (2003) 281–296.
[32] J.J. Palacios, I. González-Rodríguez, C.R. Vela, J. Puente, Robust multiobjective optimisation for fuzzy job shop problems, Appl. Soft Comput. 56 (2017) 604–616.
[33] B. Liu, Y.K. Liu, Expected value of fuzzy variable and fuzzy expected value models, IEEE Trans. Fuzzy Syst. 10 (4) (2002) 445–450.
[34] M. Sakawa, R. Kubota, Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms, European J. Oper. Res. 120 (2) (2000) 393–407.
[35] M.A. Potter, K.A.D. Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29.
[36] L.M. Antonio, C.A.C. Coello, Use of cooperative coevolution for solving large scale multiobjective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 2758–2765.
[37] L. Wang, S. Wang, Y. Xu, G. Zhou, M. Liu, A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem, Comput. Ind. Eng. 62 (4) (2012) 917–926.
[38] I. Ribas, R. Companys, X. Tort-Martorell, An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem, Expert Syst. Appl. 121 (2019) 347–361.
[39] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Springer, Netherlands, 2002.
[40] C. Wu, L. Wang, A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system, J. Parallel Distrib. Comput. 117 (2018) 63–72.
[41] J.N. Shen, L. Wang, S.Y. Wang, A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion, Knowl.-Based Syst. 74 (2015) 167–175.
[42] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European J. Oper. Res. 177 (3) (2007) 2033–2049.
[43] K.C. Ying, S.W. Lin, S.Y. Wan, Bi-objective reentrant hybrid flowshop scheduling: an iterated Pareto greedy algorithm, Int. J. Prod. Res. 52 (19) (2014) 5735–5747.
[44] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423.
[45] N. Breslow, A generalized Kruskal–Wallis test for comparing k samples subject to unequal patterns of censorship, Biometrika 57 (3) (1970) 579–594.
[46] B.B. Li, L. Wang, B. Liu, An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling, IEEE Trans. Syst. Man Cybern. A 38 (4) (2008) 818–831.
[47] D.C. Montgomery, Design and Analysis of Experiments, Vol. 52, John Wiley & Sons, 2001, pp. 218–286.
[48] H. Wang, Y. Fu, M. Huang, G.Q. Huang, J. Wang, A NSGA-II based memetic algorithm for multiobjective parallel flowshop scheduling problem, Comput. Ind. Eng. 113 (2017) 185–194.
[49] S. Hasija, C. Rajendran, Scheduling in flowshops to minimize total tardiness of jobs, Int. J. Prod. Res. 42 (11) (2004) 2289–2301.