Neurocomputing 419 (2021) 80–96

Contents lists available at ScienceDirect

Neurocomputing

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

Cooperative hybrid evolutionary algorithm for large scale multi-stage multi-product batch plants scheduling problem

Yuxin Han, Xingsheng Gu *

Key Laboratory of Advanced Control and Optimization for Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai, China

ARTICLE INFO

Article history:
Received 13 February 2020
Revised 26 May 2020
Accepted 25 July 2020
Available online 8 September 2020
Communicated by Bo Shen

Keywords:
Cooperative co-evolutionary framework
Hybrid evolutionary algorithm
Multi-stage multi-product batch plants scheduling
Large scale optimization

ABSTRACT

As an important part of batch chemical industry scheduling problems, the multi-stage multi-product batch plant scheduling problem (MMSP) has been widely studied for decades. This problem is characterized by multiple stages with non-identical parallel units and operate based on customer orders. In this paper, we focus on the large scale MMSP and treat the minimization of make-span as the objective function. An efficient cooperative hybrid evolutionary algorithm is proposed based on the framework of cooperative co-evolution. First, a novel two-line encoding scheme is developed to represent the unit assignment and sequencing for orders respectively. Second, modified estimation of distribution algorithm (EDA) and differential evolutionary (DE) operations are proposed according to the feature of MMSP. EDA operation with a novel population-based incremental learning strategy is applied to handle the unit assignment variables. And novel DE operation based on a novel encoding method is adopted to deal with sequence variables. Then, two selection strategies are applied to preserve optimal and sub-optimal solutions for the proposed algorithm. The critical path based local search algorithm is adopted to further improve the efficiency of local optimization. The proposed algorithm has been tested by several instances with different sizes and characteristics. The numerical results and comparisons show that the proposed work is very competitive in solving large scale MMSP.

1. Introduction

The multi-stage multi-product batch plant scheduling problem (MMSP) is an important part of the batch chemical industry scheduling problems, and it has received increasing attention in the past two decades. Optimal short term scheduling of such plants is formidable because of alternative units for each order in each stage. And the production costs and organization benefits are affected by scheduling schemes. Therefore, developing good schedules for batch plant is very important. Different from the multi-purpose batch plants, multi-stage multi-product batch plants have multiple stages, and each stage has parallel production units. Such plants operate based on orders' set with different batch sizes and due dates. The classic multi-stage multi-product batch plant topological structure is shown in Fig. 1. Customer orders are processed through production stages in sequence with the same recipe. If the batch size of each order keeps unchanged during the scheduling horizon, MMSP can be treated as a simplified flexible job-shop scheduling problem (FJSP).

Most of the methodologies for handling MMSP are focusing on solving mixed-integer linear programming (MILP), constraint programming (CP), heuristic, and meta-heuristic algorithms. As reviewed by Méndez et al. [1] and Franco et al. [2], MILP is still the most investigated formulation method. The majority of the MILP methods are consist of solving a discrete-time and continuous-time based formulations. Discrete-time method partition the time axis of each production unit into uniformly time slots [3]. Continuous-time models distribute time slots asynchronously for different production units and production tasks. It is first introduced by Pinto and Grossmann [4], which can handle the MMSP successfully. Afterward, the study on this problem is extended by imposing pre-ordering heuristic to reduce the computational time [5]. Another widely studied MILP formulation is the precedence-based formulation method. Gupta and Karimi [6] presented a novel MILP model that considers set times and release times of orders and units. Marchetti and Cerdá [7] introduced a new concept on this problem, namely the bottleneck stage. Based on this concept, a constant batch ordering rule (CBOR)-based formulation method is proposed, which result in the reduction of variables and superior solutions. Castro et al. [8] proposed a decomposition approach and based on the time-and sequence-based model to handle the large scale MMSP. The modeling method is further improved by

* Corresponding author.
E-mail address: xsgu@ecust.edu.cn (X. Gu).

https://doi.org/10.1016/j.neucom.2020.07.094
0925-2312/

Appendices

Indexes

i, i'	Order;
j	Unit;
s, s'	Stage;

Variables

Z_{ij}	1, order i is processed on unit j; 0, otherwise;
ZF_{ij}	1, order i is processed first on unit j; 0, otherwise;
$X_{ii's}$	1, order i' is processed after order i on same unit in stage s; 0, otherwise;
T_{is}	start time of order i in stage s;
MS	make-span;

Sets

I	Set of orders;
J	Set of units;
S	Set of stages;
J_{is}	Units set that process order i in stage s;
I_j	Orders set processed in unit j;
I_s	Orders set processed in stage s;
S_i	Stages process order i;
ns_{is}	next stage of stage s for order i;
fs_i	first stage of order i;
ls_i	last stage of order i;

Parameters:

PT_{ij}	processing time of order s on unit j;
UR_j	Unit j release time;
OR_i	Order i release time;
DD_i	Order i due date;
M	Big M value;

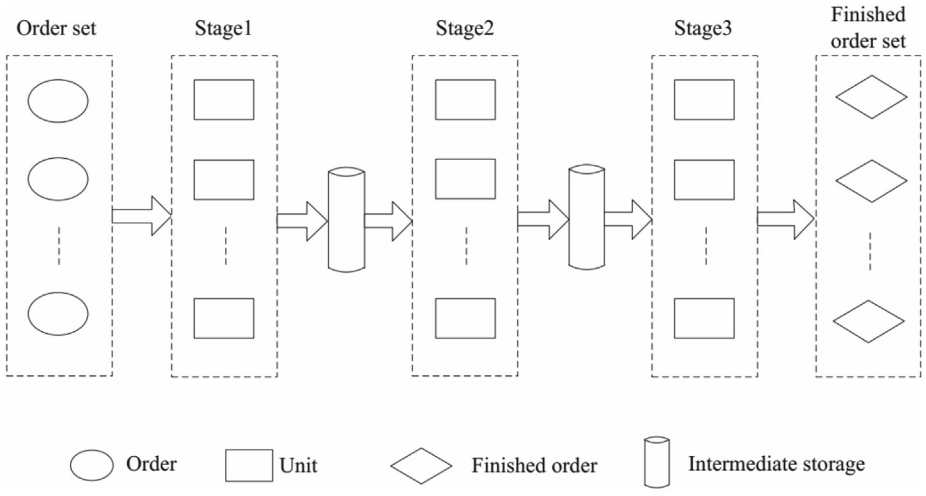


Fig. 1. Topological structure of MMSP.

considering changeover times [9]. Comparing with MILP formulation, CP is more efficient in solving minimizing the make-span [10]. Zeballos et al. [11] proposed a CP model and a search strategy that can handle MMSP with various features. Novara et al. [12] introduced a novel CP approach with the operating campaign condition. They further studied by taking the number limits of batches into account [2]. Except for MILP and CP based methods, there are many heuristic and meta-heuristic methods that been studied and designed for MMSP. He and Hui [13] address the scheduling problem by integrating the multiple heuristic rules into the genetic algorithm (GA). The search efficiency is rapidly increased by heuristic rules, and multiple rules help the GA keep the diversity. Shi and Yan [14] focused on a single-stage multi-product scheduling problem in batch plants with parallel units. They presented a novel heuristic rule-based scheduling method for solving MMSP. The lineup competition algorithm (LCA) was used to optimize order assignment rule among a group of order assignment rules. Each rule is assigned to select a suitable processing unit for each order. Later, they further extend the work to MMSP [15]. Similar to the previous work, LCA is applied with the optimal heuristic rules which can optimize MMSP.

As mentioned above, MMSP can be treated as a particular case of FJSP, which has been investigated through many theories and experiments. In this paper, the methodology of the proposed algorithm is based on the meta-heuristic algorithms that applied for FJSP. In recent years, this problem has been attracting searchers' attention, and various optimization approaches are proposed. Comparing to other types of job-shop scheduling problems, FJSP is more close to the production environment, which plays an important role in scheduling problems [16]. Meta-heuristic algorithms are effective approaches to this problem [17]. Vilcot and Billaut [18] introduced two tabu algorithms to solve a multi-criteria FJSP from the printing and boarding industry. Na and Park [19] deal with a scheduling problem with multi-level job structures in a flexible job-shop environment. GA heuristics are applied with priority rules, and results outperformed MILP and CP optimizers. A modified discrete differential algorithm (DE) is introduced by Yuan and Xu [20]. By applying forward and backward conversion techniques, operation sequence and machine assignment vectors are both could evolve with DE algorithms. A critical-operations moving technique is also improvised to improve the efficiency of the whole algorithm. Shao et al. [21] proposed a hybrid discrete particle swarm optimization (DPSO) and simulated annealing algorithm (SA), where SA is used for local search in the framework of DPSO. Gao et al. [22] adopt a novel discrete harmony search algorithm for flexible job-shop scheduling problem. Wang et al.' [23] work

is applying the EDA with bi-population to solve the FJSP. And the machine assignment and operation sequence vectors evolve in two different population respectively. Ricardo and Arturo [24] introduced a generalized Mallows distribution based EDA to solve the FJSP with process plan flexibility. Other types of FJSP is also important researching area for researchers, such as local optimization searching strategies, multi-objective problems, and distributed scheduling problems. Amiri et al. [25] focused on the neighborhood structure of the FJSP and proposed a novel variable neighborhood search algorithm. They employed a shake procedure among the several neighborhood structures to perform the local optimization search strategies. Gao et al. [26] developed a novel hybrid genetic algorithm with a variable neighborhood descent strategy. Two local search procedures are performed to handle find assignable time intervals for the deleted operations. In order to improve the focus on the efficiency of local search, Oleh and Lars [27] solved the FJSP with the objective of total weighted tardiness. The novel iterative local search heuristic hybridized shifting bottleneck heuristic with a variable neighborhood search approach. Nicoara et al. [28] solved the multi-objective problems and reinforced the NSGA_II algorithm with a heuristic adaptive control strategy to fit more optimization problems. Jian et al. [29] addressed the multi-objective FJSP with a novel hybrid evolution approach. A modified crowding distance measure based on new chromosome representation and genetic operators is proposed to preserve the diversity of the population. Lu et al. [30] put attention on the distribute job-shop scheduling problem, and proposed a novel GA-JS algorithm. They developed a new chromosome representation by converting the 3-dimensional problem into a 1-dimensional scheme. The results of GA-JS show good performance in different cell environments. De and Pezzella [31] proposed an improved genetic algorithm with a concise chromosome representation considering the distributed manufacturing environment. The algorithm considers routing, sequencing, and job assignment problems simultaneously.

In this paper, we focus on the optimization of large scale MMSP. The standard and traditional meta-heuristic algorithms cannot be applied for large scale optimization scheduling problems. Because, as the problem scale and dimension increasing, the solution space and computational cost increase exponentially, which leads to unacceptable computing time. Motivated by the reasons, there are lots of studies are carried out to handle large scale problems. The cooperative co-evolutionary algorithm (CC) is an efficient framework for solving large scale optimization problems. CC algorithm can be classified into a decomposition algorithm, which is firstly proposed by Potter and De Jong [32]. It involves a static grouping strategy that decomposed all decision variables into various groups with an identical strategy. Frans and Andries [33] presented a cooperative algorithm CPSO-HK integrating particle swarm optimizer. CPSO-SK algorithm and PSO algorithm perform alternately in the evolving process to prevent trapping into suboptimal locations in searching space. As reported by the sub-sequent researches, grouping strategy is critical to the CC. Yang et al. [34] proposed a new CC framework, which could adapt the grouping scheme automatically by weighting strategies. Omidvar et al. [35–37] carried out a series of work on the grouping strategies. They proved that, as the number of interacting variables increasing, the probability of grouping interacting variables in one subpopulation would reduce significantly. And they also proved the inefficiency of the adaptive weighting technique. A novel technique for self-adaptation of the subcomponent sizes is demonstrated for cooperative evolution [35]. Later introduced a novel technique, namely the delta method. The interacting variables are divided into equal size component by the variation of each generation. If two variables have relatively small delta value among all the variables, they are put in the same component [36]. They also proposed an automatic decomposition strategy without prior knowledge, that grouping strategy is decided by differential condition [37]. Chen et al. [38] introduced a cooperative coevolutionary algorithm with a variable interaction learning strategy. They proposed a learning stage for the algorithm, which is used to detect the search space between decision variables. And then, these groups are optimized according to the classic CC framework. Li and Yao [39] proposed a CC particle swarm optimization algorithm to solve large scale problems. Random grouping strategy is introduced in their work, which decided the coevolving subcomponent dynamically. The algorithm outperforms the state of art algorithms, and shows highly competitive search efficiency. Recently, Lu et al. [40] optimized the FJSP by proposing a distributed cooperative algorithm (dcEA). The framework is built on the resilient distributed data set (RDD). And a hybrid of GA and PSO operations is used for the evolutionary process in each RDD. Many studies of CC algorithms have shown a great advantage in solving large scale non-separable function optimization problems and sequencing optimization problems.

As introduced above, there are several challenges for solving the large scale MMSP. First is that classic evolutionary operation cannot be applied to MMSP for the unique feature. Though MMSP and FJSP are very similar, there are still several unique features of MMSP that should be taken into consideration. As shown in Fig. 1, production units are stage-specific, which means that each unit only belongs to a single stage. Recipes of orders are all the same, and operations for orders can be classified according to stages. Moreover, various constraints also greatly affect the search efficiency of meta-heuristic algorithms, such as order due date constraints, topological constraints, and sequencing constraints. These features make schedules of MMSP following particular heuristic rules. And the searching efficiency of evolutionary algorithms is also highly influenced by these features. So the encoding scheme and evolutionary operations are important to improve the search efficiency of meta-heuristic algorithms. Second, for large scale optimization problems, increasing solution space results in numbers of local optimization. Grouping strategy based CC framework has been proved to be an efficient optimization method in solving large scale problems. However, they are not applied in large scale MMSP yet. The grouping strategy and evolutionary operations for non-separable functions optimization problems need to be modified to solve this problem. To deal with these challenges, a cooperative hybrid evolutionary algorithm (CHEA) is designed in this paper to handle the large scale MMSP. The contribution of this work is listed as follows:

- A novel cooperative hybrid evolutionary algorithm with the set-based grouping strategy is designed. A grouping strategy that decomposes high dimensional decision variables into lower dimensions sub-problems is proposed based on a novel encoding scheme.
- Hybrid individual assignment and selection strategies are proposed based on the CC framework. Two archives are introduced with different selection strategies which can improve the diversity of the population.
- Two modified evolutionary operations based on EDA and DE are adopted to handle the unit assignment and sequence of orders respectively, which also can balance the ability of exploration and exploitation effectively.

Experimental studies demonstrate the effectiveness and efficiency of the proposed algorithm in solving large scale MMSP. The organization of this paper is listed as follows: Section 2 introduces the formulation of MMSP. Section 3 introduces the detailed improvement of CHEA algorithm. Section 4 shows the results of

experiments and analysis. Section 5 gives the conclusions and future work.

2. Problem description and formulation

Formulation methods for MMSP are widely investigated by many researchers, and many formulation methods have been proposed. As shown in Fig. 1, customer orders are known in advance before the production process, and each order represents a single product. The batch size of each order keeps unchanged during the scheduling horizon. Each order requires a due date and release time. There are a set of non-identical processing units that belong to a single stage, and each unit could only be operated in a single stage. The objective function of minimizing make-span is pursued in this paper. Additionally, forbidden unit assignment, orders due date timing, and unlimited intermediate storage constraints are also taken into consideration for MMSP. To describe MMSP precisely in this work, the mathematical formulation of MMSP from [6] is introduced in Eqs. (1)–(12). The definition of indexes, variables, sets, and parameters are listed in appendices.

Eq. (1) represents the order assignment constraints, which means that each order i can only be processed on a single unit j in stage s.

$$\sum_{j\in \mathrm{J}_{is}} Z_{ij} = 1, i \in \mathrm{I}_j \tag{1}$$

Eqs. (2)–(4) indicate orders sequence on each unit. Eqs. (2) and (4) indicate that only one order can be the first order on each unit j. Eq. (5) represents the sequence constraint for different orders i and i' on the same unit j.

$$\sum_{i\in \mathrm{I}_j} ZF_{ij} \le 1 \tag{2}$$

$$\sum_{i' \in \mathrm{I}_s} X_{ii's} + \sum_{j\in \mathrm{J}_{is}} Z_{ij} = 1, i \in \mathrm{I}_s \tag{3}$$

$$Z_{ij} \ge ZF_{ij}, i \in \mathrm{I}_j \# \tag{4}$$

Eqs. (5) and (6) are unit assignment constraints, which decide the unit assignment for different orders on the same unit. If integer variable $X_{ii's}$ or $X_{i'is}$ is activated, order i' and i must be processed in the same unit j.

$$2\left(X_{ii's} + X_{i'is}\right) + \sum_{j\in \mathrm{J}_{is}-\mathrm{J}_{i's}} Z_{ij} + \sum_{j\in \mathrm{J}_{i's}-\mathrm{J}_{is}} Z_{i'j} \le 2, i' > i, \left(i', i\right) \in \mathrm{I}_s \tag{5}$$

$$Z_{ij} \le Z_{i'j}, j \in \mathrm{J}_{is} \cap \mathrm{J}_{i's}, i' > i, \left(i', i\right) \in \mathrm{I}_s \tag{6}$$

Order timing constraints are presented in Eqs. (7)–(9). Eq. (7) shows timing constraints for a single order in different stages. Eq. (8) is the timing constraint for different orders on the same unit. If unit release time UR_j or order release time OR_i are considered, (9) and (10) are considered. Eq. (11) represents the constraint for cases with due date DD_i.

$$T_{is'} \ge T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}\mathrm{PT}_{ij}, s' = \mathrm{ns}_{is}, s \in \mathrm{S}_i \tag{7}$$

$$M\left(1 - X_{ii's}\right) + T_{i's} \ge T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}\mathrm{PT}_{ij}, s' = \mathrm{ns}_{is}, s \in \mathrm{S}_i \tag{8}$$

$$T_{is} \ge \sum_{j\in \mathrm{J}_{is}} ZF_{ij}\mathrm{UR}_j, i \in \mathrm{I}_s \tag{9}$$

$$T_{is} \ge \mathrm{OR}_i, s = \mathrm{fs}_i \tag{10}$$

$$T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}t_{ij} \le \mathrm{DD}_i, i \in \mathrm{I}, s = \mathrm{ls}_i \tag{11}$$

In this work, the objective function of minimizing make-span is adopted to evaluate the performance of the proposed algorithm, as shown in Eq. (12).

$$\mathrm{MS} = min\left(\max_i \left(T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}\mathrm{PT}_{ij}\right)\right), s = \mathrm{ls}_i \tag{12}$$

It is worthy to note that, all the constraints can be easily satisfied except Eq. (11) for minimizing make-span by meta-heuristic algorithms. To fit in the Eq. (11), the penalty function method is adopted for the proposed algorithm. And Eqs. (13) and (14) are used to calculate the fitness in this paper.

$$d_i \ge max\left(T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}\mathrm{PT}_{ij} - \mathrm{DD}_i, 0\right), i \in \mathrm{I}, s = \mathrm{ls}_i \tag{13}$$

$$\mathrm{fitness} = min\left(\max_i \left(T_{is} + \sum_{j\in \mathrm{J}_{is}} Z_{ij}\mathrm{PT}_{ij}\right)\right) + \mathrm{M}\cdot\sum_{i\in \mathrm{I}} d_i, s = \mathrm{ls}_i \tag{14}$$

Eq. (13) defines the punishment function and the big-M method is applied to punish the violation in Eq. (14). If the finish time of each order is beyond the corresponding due date, Eq. (13) is activated. And the fitness value in (14) would become deteriorate.

3. Cooperative hybrid evolutionary algorithm

The CHEA is introduced in detail in this section. To show the scheme of this work more clearly, the flow chart of CHEA is shown in Fig. 2.

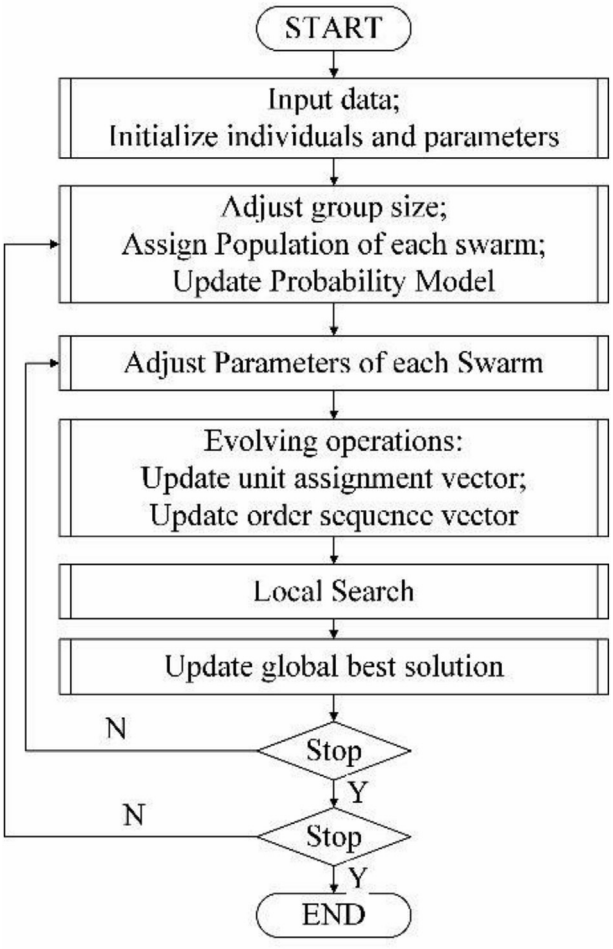


Fig. 2. Flow chart of CHEA.

3.1. Solution representation method

There are two problems that should be considered simultaneously for MMSP: unit assignment and sequence of orders. The sequence of each job on each production unit is usually defined as a variable. In this paper, to improve the search efficiency and cooperate with the evolutionary operation, two sub-problems are represented separately similar to two vectors representation method, as shown in Eqs. (15)–(18). Set $\mathbf{P}_r$ represent the solution of individual r in the population. Each subset $\mathbf{P}_{r,i}$ of set $\mathbf{P}_r$ contains all the unit assignment and order sequence of single order i in all stages. Set $\mathbf{P}_{r,i}$ includes two vectors, i.e. unit assignment vector $\mathbf{Pu}_{r,i}$ and sequence vector $\mathbf{Pv}_{r,i}$. Unit assignment vector $\mathbf{Pu}_{r,i}$ is constructed by integer variables $u_{r,i,s}$, which represent that order i is processed on unit j in stage s. And continuous variables $v_{r,i,s}$ in vector $\mathbf{Pv}_{r,i}$ represent the sequence value of order i in stages. It is worth to note that, vectors $\mathbf{Pu}_{r,i}$ and $\mathbf{Pv}_{r,i}$ contain all variables of all stages belong to a single order i. And each set $\mathbf{P}_{p,i}$ contains all the variables belong to a single order.

$$\mathbf{P}_r = \{\mathbf{P}_{r,i}, i \in Orders\}, p \in Population \tag{15}$$

$$where, \mathbf{P}_{r,i} = \{\mathbf{Pu}_{r,i}, \mathbf{Pv}_{r,i}\} (\tag{16}$$

$$\mathbf{Pu}_{r,i} = \left[u_{r,i,s1}, u_{r,i,s2}, \cdots, u_{r,i,|S|}\right], i \epsilon I \tag{17}$$

$$\mathbf{Pv}_{r,i} = \left[v_{r,i,s1}, v_{r,i,s2}, \cdots, v_{r,i,|S|}\right], i \epsilon I \tag{18}$$

where $|*|$ represent the size of set, and $|S|$ means the number of stages. Orders' sequence on each production unit is sorted based on the value of $v_{r,i,s}$. The lower value has higher priority, and higher value has a lower priority, which is the same as the random key representation method. However, the feasible search space of continuous variables is too large that might slow down the search steps. To improve search efficiency, the sequencing variables are calculated by the Eq. (19):

$$v_{r,i,s} = st_{r,i,s} + a \cdot rand \cdot \mathrm{PT}_{u^*}, u^* = u_{r,i,s} \tag{19}$$

where,$st_{r,i,s}$ is the start time of order i in stage s, PT_{u^*} is the processing time of order i on the corresponding unit u^* in stage s. Parameter a is the scaling factor, and$rand$ is a random value belong to (0,1). The first term on the right side of Eq. (19) represents that, the sequence variables are decided based on scheduling time. Thus, the feasible solution space of continuous variable $v_{r,i,s}$ is reduced to discrete values, i.e. start time of each order in scheduling horizon. The scaling factor a and random value $rand$ in the second term are to avoid the same value in the DE operation. For the same value of sequence variables might go wrong during the sorting procedure. The parameter PT_{u^*} here is to restrict the random value into the scheduling time dimension. Parameter a controls the influence of the PT_{u^*}. In this paper, a is set to a small value that makes the second term as small as possible.

To state the encoding method more clearly, an example with 6 orders, 4 units, and 2 stages is shown in Fig. 3. The unit assignment vectors and order sequence vectors of 2 stages are listed in the figure. Four orders are processed on unit 1 in stage 1, i.e. order $i1$, $i3$, $i4$, and $i5$. Sorting the sequence variables $v_{r,i1,s1}$, $v_{r,i3,s1}$, $v_{r,i4,s1}$ and $v_{r,i5,s1}$ from smallest to largest, and the production sequence on unit 1 is $i5 \rightarrow i1 \rightarrow i3 \rightarrow i4$. By repeating the process, we can obtain orders' sequences of other stages as shown in Fig. 3.

3.2. Cooperative co-evolutionary framework

The framework of cooperative co-evolution is introduced to handle large scale MMSP in this work. CC algorithm decomposes a large scale problem into sub-problems by decomposing the decision variables to different groups. Each group constructs a swarm that evolves in parallel. And the numerical results greatly proved the optimization abilities of this framework. In this paper, the set-based random grouping decomposition strategy is adopted, which is widely used for optimizing the large scale problems.

Based on the novel encoding scheme, each set $\mathbf{P}_{r,i}$ is defined as the decision set, which equivalent to decision variables in the classic CC framework. Each set contains the unit assignment vector $\mathbf{Pu}_{r,i}$ and order sequence vector $\mathbf{Pv}_{r,i}$ of one order i in all stages. So when the set-based random grouping decomposition strategy is performed, set $\mathbf{P}_{r,i}$of all orders are decomposed into groups. For example, as depicted in Fig. 4, we consider the instance with 12 orders, and the group sets are K = 3 and K = 4. If the K = 3group is set, 12 variables are decomposed into 3 groups with 4 variables of each. When K = 4, 4 groups are constructed, each composed of 3 randomly dispatched sets. Each group constructs a swarm with m individuals and perform the evolutionary process in parallel.

While evaluating each individual in each swarm, the original individual returns a new individual by replacing its jth variable in each group. As shown in Fig. 5, we taking K = 4as an example. After grouping and evolutionary operation, newly generated sets $\mathbf{P}_{r,i1}'$, $\mathbf{P}_{r,i3}'$,$\mathbf{P}_{r,i6}'$and$\mathbf{P}_{r,i11}'$replace the original variables $\mathbf{P}_{r,i1}$,$\mathbf{P}_{r,i3}$, $\mathbf{P}_{i,i6}$ and $\mathbf{P}_{r,i11}$ in individual R1. Thus, a new individual R1′ is generated.

In this work, we adopt the randomly grouping strategy, in which the group size K of decision sets is adjusted in each iteration. The group size of each iteration is selected from the grouping set. As depicted in Fig. 4, considering the case with group set $S_K = \{3, 4\}$, if the best solution is not updated, the value of K choose from the S_K randomly.

The CC framework introduced above is applied based on the novel encoding method in this paper. The classic cooperative coevolutionary algorithm is proposed based on the idea that grouping related variables into in the same group. However, as the number of decision variables increases, the probability of grouping interacting variables in one subpopulation would reduce significantly [35]. The timing constraint in Eq. (7) indicates that the sequence variables in different stages of an order are related variables. So the encoding scheme proposed in this work divide the variables that closely related to a single order into one sub-group $\mathbf{P}_{r,i}$ in advance. Then the random grouping strategy can only focus on constructing the relationship between decision sets. Thus, the searching efficiency can be highly improved by avoiding large amounts of invalid grouping operations in the CC algorithm.

3.3. Assignment and selection operation

In the classic CC algorithm, individuals in each sub-population can only replace the grouped decision variables in the best solution of each generation. And the alternative choice for each individual is highly limited. To increase the search efficiency and improve the diversity of CC framework, the optimal solutions and sub-optimal solutions are both taken into consideration. Two archives that represent different individual selection strategies are proposed here. The first archive, namely archive **A**, selects the best individuals by classic selection operation. The second achive, namely archive **B**, saves the suboptimal solutions of the population in each generation. And two archives are performed through assignment and selection operations in this work.

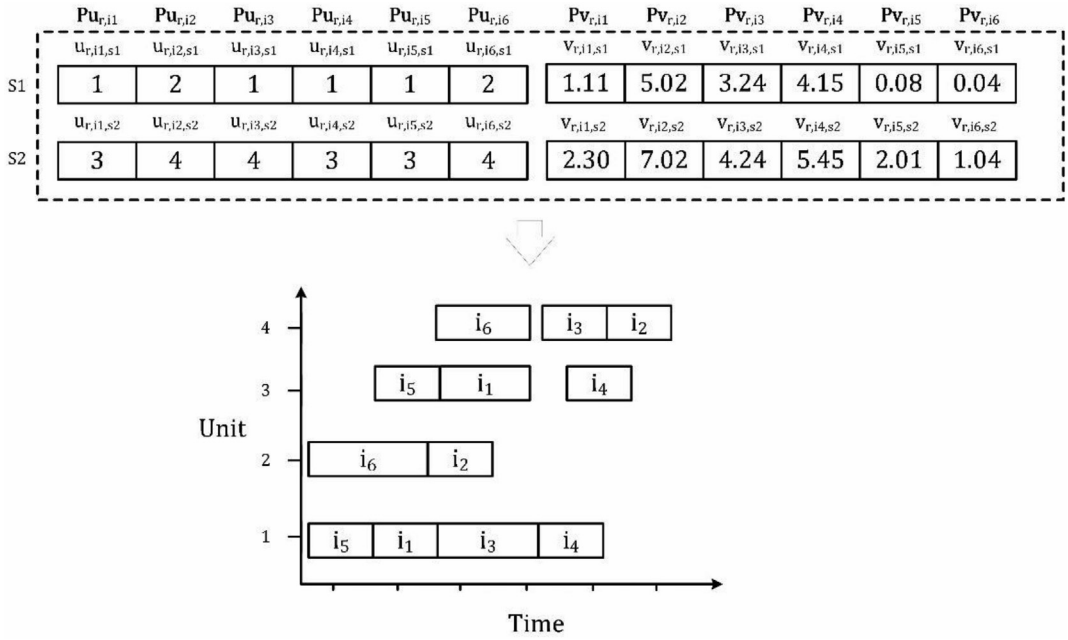


Fig. 3. Illustration of the solution representation method.

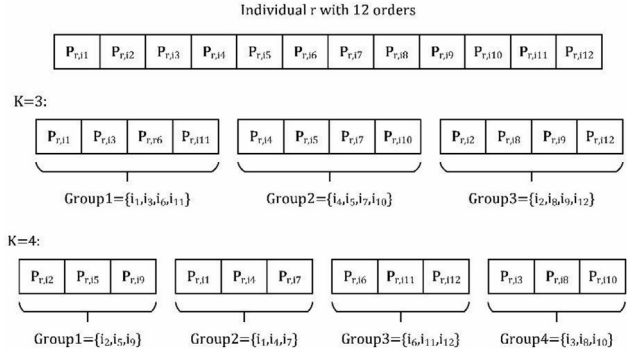


Fig. 4. Illustration of set-based grouping strategy.

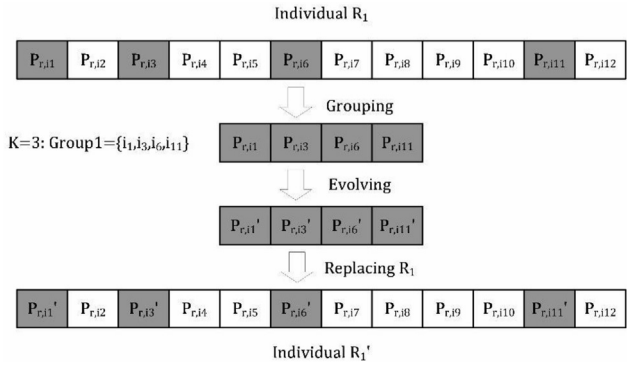


Fig. 5. Example of evaluating the individual R1.

3.3.1. Assignment operation for archives

The assignment operation is necessary, and the population is constructed by individuals from both archives. The pseudocode of assignment operation is shown in algorithm 1.

Algorithm 1: Assignment Operation

Input: archive **A**, archive **B**;
Output: each swarm $\mathbf{P}_k$;

```
for each swarm k do
  for each individual r_k^P in swarm P_k do
    if rand < p_A
      P_{k,r_g^P} ← A_{r^A}, r^A ← rand(1, |A|);
      Sp(r_k^P) ← r^A;
    else
      P_{k,r_k^P} ← B_{r^B}, r^B ← rand(1, |B|);
      Sp(r_k^P) ← r^B;
    end if
  end for
end for
```

Where $r_g^{\mathbf{P}}$,$r^{\mathbf{A}}$ and $r^{\mathbf{B}}$ are individual number of population $\mathbf{P}_g$, archive **A** and archive **B** respectively of generation g. $|\mathbf{A}|$ and $|\mathbf{B}|$ means the size of archive **A** and **B**. *Sp* is the array that preserves the exact individual chosen from two archives for each individual in the population. $p_{\mathbf{A}}$ is the probability that chosen from archive **A**, and $p_{\mathbf{A}}$ is set to 0.5 in this paper. With algorithm 1, the population of each swarm is constructed by dispatching the individuals from both archives **A** and **B**. And the fitness of each individual in each swarm is evaluated by replacing the grouped decision sets in assigned solution from both archives.

Assigning optimal and sub-optimal individuals to the population can improve the diversity of population. Because the evolving procedure starts based on the best solutions at each generation. Here in this work, the size of archive **A** and **B** follow the rules below:

$$|\mathbf{A}| \le \left(\sum_{g\in G}|\mathbf{P}_g|\right) * p_{\mathbf{A}} \tag{20}$$

$$|\mathbf{B}| \le \left(\sum_{g\in G}|\mathbf{P}_g|\right) * (1 - p_{\mathbf{A}}) \tag{21}$$

Eqs. (20) and (21) means that each individual in archives has the opportunity to be allocated into the evolving population. And some of the individuals in archives might be assigned more than once. If $p_{\mathbf{A}}$ equals 1, only best individuals could enter the next generation, which is the same as the classic CC algorithm.

3.3.2. Selection operation

Classic selection operations that widely used in the evolutionary algorithms are modified to fit in the proposed algorithm as introduced below. In the classic DE algorithm, the selection operation only performed in between the trial vector and current vector, as shown in Eq. (22):

$$\mathbf{u}_{r1}^{t} = \begin{cases} \mathbf{v}_{r1}^{t}, if\ fitness(\mathbf{v}_{r1}^{t}) < fitness(\mathbf{u}_{r1}^{t-1}) \\ \mathbf{u}_{r1}^{t-1} \end{cases} \tag{22}$$

where the trial vector $\mathbf{v}_{r1}^{t}$ is chosen into the next generation only if get the fitness is better. This method has been proved to an efficient method for the DE algorithm. However, this selection method might miss sub-optimal solutions. So in this work, sub-optimal solutions can be saved in archive **B**, and archive **A** is only used for saving optimal solutions selecting by Eq. (22). The selection operations for archives **A** and **B** are shown in algorithm 2.

Algorithm 2: Selection Operations of archives A and B

Input: archive $\mathbf{A}^{t-1}$, archive $\mathbf{B}^{t-1}$, population of all swarms $\mathbf{P}_k^{t-1}$;
Output: archive $\mathbf{A}^{t}$, archive$\mathbf{B}^{t}$

%update archive$\mathbf{A}^{t}$
1: **for** *each swarm k*
2: **for** *each individual* $r_k^{\mathbf{P}}$ *in population* $\mathbf{P}_k^{t-1}$ **do**
3: $r^{*} \leftarrow Sp(r_k^{\mathbf{P}})$
4: **if**fitness$(\mathbf{P}_{k,r_{\mathbf{P}}}^{t-1})$ < fitness$(\mathbf{A}_{r^{*}}^{t-1})$
5: $\mathbf{A}_{r^{*}}^{t} \leftarrow \mathbf{P}_{k,r_{\mathbf{P}}}^{t-1}$;
6: $\mathbf{P}_k^{t-1} \leftarrow \mathbf{P}_k^{t-1} - \mathbf{P}_{k,r_P}^{t-1}$;
7: **else**
8: $\mathbf{A}_{r^{*}}^{t} \leftarrow \mathbf{A}_{r^{*}}^{t-1}$;
9: **end if**
10: **end for**
11: **end for**
%update archive $\mathbf{B}^{t}$
12: $\mathbf{P}_0^{t} \leftarrow (\underset{k\in K}{\cup} \mathbf{P}_k^{t-1})$;
13: **for** $r_k^{B} \leftarrow$ from 1 to $|\mathbf{B}|$ do
14: $\mathbf{B}_{r_k^{\mathbf{B}}} \leftarrow$ best fitness individual r^{*} in $\mathbf{P}_0^{t}$;
15: $\mathbf{P}_0^{t} \leftarrow \mathbf{P}_0^{t} - \mathbf{P}_{r^{*}}^{t}$;
16: **end for**

Where lines 1 ~ 11 represent the selection operation of archive **A**, and lines 12 ~ 16 are for archive **B**. Lines 6 and 15 mean that a selected individual should be removed from the population set. An individual cannot be selected twice in each generation. The selection operation of archive **A** is the same as classic DE selection operation. And archive B is only used for saving sub-optimal solutions.

Moreover, while choosing individuals for archive **B** in algorithm 2 line 14, the fitness of different individuals are equal in some cases. To avoid this situation, a multiple objectives strategy should be applied in some cases just for selecting different sub-optimal individuals. For example, when considering the objective make-span, different individuals might have the same fitness. However, equal make-span might lead to different scheduling solutions. To handle this problem, another objective function, namely total flow time is adopted in this paper to distinguish the difference between individuals. If two individuals have the same make-span, the one with smaller total flow time can be selected into archive **B**. In this way, the diversity of the population can be easily improved.

In this paper, the diversity of population can be ensured by two archives with different selection methods. Since there are alternative choices for each individual. Not only best offspring can be chosen for each individual by selecting archive **A**. But the sub-optimal offspring have the opportunity to be passed to the next generation by archive **B** as well. Two archives represent two different selection methods, which can get best solutions from different perspectives. Through assignment operation, grouped variables have more evolutionary direction for each grouped variables from different solutions. Thus, the diversity of the population of the CC algorithm can be further improved. Moreover, when $|\mathbf{A}| = 1, |\mathbf{B}| = 0$, or $|\mathbf{A}| = 0, |\mathbf{B}| = 1$, the framework of this paper can be treated as a classic CC algorithm. It noteworthy that, two selection operations are widely used in other meta-heuristic algorithms, such as GA, PSO, and so on. However, different from classic meta-heuristic algorithms, selection operations in this paper select optimal and sub-optimal individuals for archives instead of population. CHEA employs the hybridization of both selection methods to save the best and sub-optimal offspring at the same time.

3.4. Evolutionary operation

3.4.1. EDA operation

The EDA has been widely studied in solving continuous and discrete problems. In Wang et al.' [23] work, both unit assignment and order sequence variables are estimated by the probability model of EDA. In this work, EDA operation is adopted to update the unit assignment vectors. There are mainly three steps for the EDA: selecting promising individuals, building the probabilistic model, and generating new candidate individuals. To improve the search efficiency, population-based incremental learning (PBIL) strategy is applied in this paper, as shown in Eq. (23):

$$p_i^{t} = p_i^{t-1} * (1.0 - \mathrm{LR}) + \mathrm{LR} * q_i^{t} \tag{23}$$

where p_i^{t-1} and p_i^{t} is the probability of generating a 1 in bit position i in generation t-1 and t respectively, q_i^{t} is the probability of ith position in the solution string and LR is the learning rate. However, classic PBIL in (23) might lead to premature and local optimization for the proposed algorithm. A modified PBIL method is proposed for dealing with large scale MMSP, as shown in Eq. (24):

$$p_{i,j}^{t} = p_{i,j}^{t-1} * (1.0 - \mathrm{LR} - \mathrm{DR}) + \mathrm{LR} * q_{i,j}^{t} + \mathrm{DR} * r_{i,j}, i \in \mathrm{I_J}, j \in \mathrm{J_S} \tag{24}$$

where $p_{i,j}^{t-1}$ and $p_{i,j}^{t}$ are the probability of assigning order i on unit j in generation t-1 and t respectively. And $p_{i,j}^{t}$ follows the constraint that:

$$\sum_{j\in \mathrm{J}_{is}} p_{i,j}^{t} = 1, s \in \mathrm{S}$$

Parameters $q_{i,j}^{t}$ is the probability of order i on unit j in the population of generation t. DR is defined as the degenerate rate within the range of [0,1], and $r_{i,j}$ is the probability with a uniform distribution of all processing units of order i in each stage. Here, $r_{i,j} = 1/|\mathrm{J}_{is}|$, and $|\mathrm{J}_{is}|$ means the number of units which can process order i in stage s. If there are n units for order i in stage s, $|\mathrm{J}_{is}| = 1/n$, i.e. $r_{i,j} = 1/n$.

To prevent premature convergence, each probabilistic vector in Eq. (23) is varied slightly which is based on the mutation rate. However, there is a special situation that must be taken into consideration for Eq. (23). If probability $q_{i,j}^t = 1$ or $q_{i,j}^t = 0$ appear consecutively for generations, the probability model of $q_{i,j}^t$ is likely to be 1 or 0. In such cases, the newly generated population might trap into local optimization. This situation is more likely to appear in the proposed algorithm, and it cannot be ignored. Thus, DR is introduced to prevent this situation. In the late period of the algorithm, we can get $p_{i,j}^t \approx p_{i,j}^{t-1}$ and $q_{i,j}^t \leq 1$. The Eq. (24) can be turned into (25):

$$p_{i,j}^t \leq p_{i,j}^t \cdot (1.0 - \mathrm{LR} - \mathrm{DR}) + \mathrm{LR} + \mathrm{DR} \cdot r_{i,j}, i \in \mathrm{I}_J, j \in \mathrm{J}_S, t \rightarrow MaxGeneration \tag{25}$$

Define $x = \mathrm{DR}/\mathrm{LR}$, Eq. (25) is simplified to (26)

$$p_{i,j}^t \leq \frac{1+x \cdot r_{i,j}}{1+x}, t \in AllGeneration, q_{i,j}^t = 1 \tag{26}$$

With the same method, if $q_{i,j}^t \geq 0$, Eq. (27) is get:

$$p_{i,j}^t \geq \frac{x \cdot r_{i,j}}{1+x}, t \in AllGeneration, q_{i,j}^t = 0 \tag{27}$$

Thus, the curve of Eqs. (26) and (27) is illustrated in Fig. 6. As shown in this figure, as the value of x increasing, the range of probability vector is getting smaller. In other words, if the learning rate LR is a constant, the larger x is, i.e. degenerating rate DR, the smaller variable range of probability vector would be. If $\mathrm{DR} = 0$, $0 \leq p_{i,j}^t \leq 1$. And if $\mathrm{DR} \neq 0$, $0 < p_{i,j}^t < 1$. When the individuals trapped into local optimization, i.e. $p_{i,j}^t = 1$ or $p_{i,j}^t = 0$, new unit assignment vectors can generate with non-zero probabilistic vectors. Thus, premature of the individual can be avoided. In this work, the value of x is decided based on trial and error as shown in the numerical experiment section.

The procedures of building the probability model and generating the unit assignment vectors are performed in different processes in the cooperative algorithm. The probabilistic model is built before decision sets are grouped. And unit assignment vectors are sampled by the probability model in each sub-population. Based on the Eq. (24) and classic EDA procedure, the pseudocode of building probability is shown in algorithm 3.

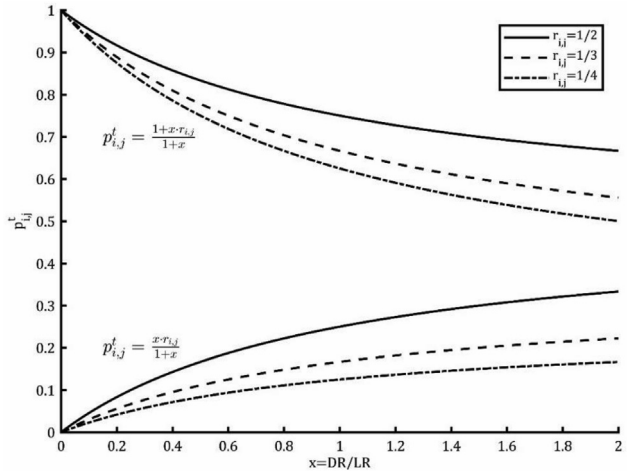


Fig. 6. Curve of Eqs. (26) and (27).

Algorithm 3: Built the probability model

Input: population $\mathbf{P}^{t-1}$ of all swarm;
Output: probability model $p_{o,u}^t$

1: select $\tau\%$ promising population $\mathbf{S}^t$ from population $\mathbf{P}^{t-1}$;
2: $n_{i,j^*} = 0$;
3: **for** each individual $\mathbf{P}_i$ in $\mathbf{S}^t$ **do**
4: **for** each order i in stage s **do**
5: $j^* = u_{r,i,s}$;
6: $n_{i,j^*} = n_{i,j^*} + 1$;
7: **end for**
8: **end for**
9: $q_{i,j}^t = \frac{n_{i,j^*}}{(0.01 * \tau * |\mathbf{P}^{t-1}|)}$
10: update probability vector with Eq. (24)

In the line 1, the selected size for $\mathbf{S}^t$ is decided by threshold parameter τ, which selects the best $\tau\%$ solutions based on fitness. The sampling operation is based on the probability model, and roulette selection mothed is adopted here to generate the unit assignment vectors. EDA operation only decides the unit assignment vectors of each individual, and it is embedded into the DE operation as introduced as follows.

3.4.2. DE operation

Classic DE algorithm consists of three parts, including mutation, crossover, and selection operations. In this paper, mutation operation is used to update the orders' sequence vectors. As introduced in the encoding method, the sequence vectors are constructed by continuous variables, which can be applied directly in the mutation operation in this paper.

Each sequence vector contains the sequence variables of a single order in different stages. Sequence variables indicating the sequence in different stages of an order are classified into two categories. First is that the variables mutate with the current-to-rand operation which is widely used in the classic DE algorithm. Second is that, other variables remain unchanged. Novel mutation operation is shown in algorithm 4.

Algorithm 4: Mutation operation

Input: vectors $\mathbf{Pv}_{r1,i}^{t-1}$, $\mathbf{Pv}_{r2,i}^{t-1}$, $\mathbf{Pv}_{r3,i}^{t-1}$ and $\mathbf{Pu}_{r1,i}^{t-1}$ of order o of each swarm
Output: vectors $\mathbf{Pv}_{r1,i}^t$, $\mathbf{Pu}_{r1,i}^t$

1: **for** *each Stage* $s \in (s_0, s_0 + s_L - 1)$ **do**
2: Perform sampling operation of EDA operation for $u_{r1,i,s}^t$ with $p_{i,j}^t$;
3: **if** *Stages* $\leftarrow s_0$
4: $v_{r1,i,s}^t \leftarrow v_{r1,i,s}^{t-1} + f_1 * (v_{r2,i,s}^{t-1} - v_{r3,i,s}^{t-1})$;
5: **else**
6: $v_{r1,i,s}^t \leftarrow v_{r1,i,s}^{t-1} + \mathrm{PT}_{r1,s-1} + f_2 * (v_{r2,i,s}^{t-1} - v_{r3,i,s}^{t-1})$;
7: **end if**
8: **end for**
9: **for** *each Stage* $s \notin (s_0, s_0 + s_L - 1)$ **do**
10: $v_{r1,i,s}^t \leftarrow v_{r1,i,s}^{t-1}$;
11: $u_{r1,i,s}^t \leftarrow u_{r1,i,s}^{t-1}$;
12: **end**

Where s_0 is the first stage that required to mutate, and s_L is the number of sequence variables that require to mutate. r_1 is the current individual, and r_1, r_3 are randomly chosen individuals from the population. f_1 and f_2 are scaling factors, and $PT_{r1,s-1}$ is the production time of r_1 in $s-1$ stage.

The mutation operation is performed cooperating with the Eq. (19) as introduced in section 3.1. All the variables are defined and mutate based on time representation, including start time and processing time. The mutation operation includes two sections to deal with two classes of variables in each sequence vector. For the changing sequence variables, lines 1 ~ 8 indicate the mutation operation. Line 4 indicates the EDA operation that generates the novel unit assignment variable $u^t_{r1,i,s}$ based on the probability vector. There are two kinds of differential equations shown in lines 4 and 6. The sequence variables in the first stage mutate by a classic differential equation, while others with the equation in line 6. The first term $(v^{t-1}_{r1,i,s} + PT_{r1,s-1})$ on the right side of the equation in line 6 means that sequence variables mutate based on the finish time of the same order in the previous stage. For the unit assignment and order sequence variables that keep unchanged is shown in lines 9–12. This mutate operation is proposed based on the encoding method. For the encoding scheme is proposed based on production time, and the finish time of each in the previous stages -1 can easily be the reference for the stage s. This method is based on the feature of MMSP that, the sequence of orders is decided by the orders' finish time in the previous stage.

There are two important parameters in the mutation operation, s_0 and s_L. For example, when the $s_0 = 1$ and $s_L = |S|$, all the variables in each sequence vector $\mathbf{Pv}^t_{r1,i}$ mutate. On the contrary, if the s_0 equals 1 and s_L equals zero, only one sequence variable mutate. s_0 is decided according to s_L, as shown in Eq. (28):

$$s_0 = rand(1, |S| - s_L + 1) \tag{28}$$

The exploration and exploitation of proposed algorithm is highly influence by DE operation in this paper. While $s_L = |S|$, the algorithm favors the ability of exploration over exploitation. Since all the variables in the decision set $\mathbf{P}_{r,i}$ participate in the evolutionary operation. On the contrary, as the value of s_L decreasing, i.e. $s_L < |S|$, the number of variables that participate in evolution also gradually decreases. In this process, the exploitation of novel algorithm is strengthened, and exploration is weakened. So in this paper, to balance the ability of exploitation and exploration, multiple s_L is applied for CHEA. The value of s_L is chosen from a predetermined set randomly, when the grouping strategy is performed during the process of evolution. And each individual only chose a single value of s_L.

There are three problems worth mentioning. First is that the parameter *rand* introduced in Eq. (19) encoding method section 3.1. It is very common that different orders have the same start time or finish time in different schedules. So if *rand* is not considered, the DE operations in lines 4 and 6 of algorithm 4 can also lead to equal value with different sequence values, which is also another sequencing problem. For example, if *rand* is not considered in Eq. (19), $v_{r,i,s} = st_{r,i,s}$. And if $st_{r2,i,s} = st_{r3,i,s}$, i.e. $v^{t-1}_{r2,i,s} = v^{t-1}_{r3,i,s}$, we can get $v^t_{r1,i,s} = v^{t-1}_{r1,i,s}$. In the same schedules, different orders might have equivalent start time value, i.e. $v^{t-1}_{r1,i,s} = v^{t-1}_{r1,i^*,s}$, where i and i^* are different orders. Equal value of sequence variables might lead to permutation problems. To avoid this problem and simplify the sequencing procedure, the parameter *rand* is introduced to get different values for sequence variables. Second problem is that, lines 9 ~ 12 indicate the situation that, while $s_L = |S|$, some variables in unit assignment and sequence vectors keep unchanged during evolutionary progress. This has a great impact on the probability model. Considering the selection and assignment operations, it increases the probability of occurrence that special situation discussed in section 3.4.1. So the degenerating parameter DR is critical in this algorithm, and it is discussed with numerical results in the numerical experiment. The last one is that, to improve search efficiency, the adaptive adjustment method for scaling factors f_1 and f_2 in *j*DE [41] are adopted in this work, as shown below:

$$f^t_l = \begin{cases} f_{up} + rand \cdot f_{lo}, \text{if } rand < \delta \\ f^{t-1}_l \end{cases}, l = 1, 2$$

where, f_{up} and f_{lo} are upper and lower bounds of f^t_l, and δ is the shifting probability.

3.5. Critical path based local search strategy

The critical path-based local search algorithm is a widely investigated local searching strategy in solving FJSP. The objective make-span is determined by the longest critical path, which contains critical operations. In this work, to further improve the local exploitation, we adopt the critical path based local searching strategy for CHEA. The algorithm of getting the critical path is shown in algorithm 5.

Algorithm 5: Get Critical Path

Input: individual **P**;
Output: critical path **G**;
1: Initialize critical path set $\mathbf{G} = \phi$;
2: Find the order $i^* = i_{s_{final}}$ with largest finish time in the final stage s_{final}, $\mathbf{G} \leftarrow \mathbf{G} \cup i^*$;
3: Stages $\leftarrow |S|$
4: **repeat**
5: find previous order i' on the same unit;
6: **if** $ft_{s-1,i^*} > ft_{s,i'}$
7: $s \leftarrow s - 1$;
8: **else**
9: $i^* \leftarrow i'$;
10: **end if**
11: $\mathbf{G} \leftarrow \mathbf{G} \cup i^*$;
12: **until** s = 1 and $i^* = \phi$

Where $ft_{s,i}$ represents the finish time of order i in stage s, and order i' is the processed before order i^* on the same unit. When $s = 1$, $ft_{s-1,i} = 0$. The corresponding local search algorithm is shown in algorithm 6. The computational cost of the local search strategy is large. So in this paper, the local search algorithm is applied with the probability of β for individuals in each swarm.

Algorithm 6: Critical path based Local Search

Input: individual P;
Output: updated individual $\mathbf{P}'$;
1: Obtain critical path set (G) of P by **Algorithm 5**;
2: **for** *each order i in* G **do**
3: Remove *o* from G to get G_0;
4: Find the feasible time interval in G_0;
5: **for** *each feasible time interval* **do**
6: Assign order *i* in the time interval to get individual $\mathbf{P}'$;
7: **if** *fitness* $\mathbf{P}'$ *is better than* **P**
8: Update solution **P**;
9: **end if**
10: **end for**

(*continued*)

Algorithm 6: Critical path based Local Search

```
    if no improvement found then
      delete i and i' from G to get G1;
      Find the assignable time intervals for i and i' in G1;
      for feasible time intervals do
      Assign order o and o' in the time interval to get individual P'';
      if fitness P'' is better than Pthen
        Update solution P;
      end if
    end for
  end if
end for
```

3.6. Framework of CHEA algorithm

As described above, the pseudocode of the CHEA algorithm employing critical path-based local search strategy is shown in algorithm 7. Lines 2 ~ 4 represent the random grouping strategy. Lines 5 ~ 8 update both archives and population. The evolutionary operation of each swarm is shown in lines 9 ~ 15.

Algorithm 7: CHEA algorithm

```
Create and initialize K swarms by randomly choose from a grouping set S.
1:  repeat
2:    if global fitness not improved
3:      Randomly choose group size s from set S_K, and calculate swarm number K;
4:      Randomly choose s_L from set S_L, and generate s_0 with Eq. (28) for each individual;
5:    end if
6:    Update archives A and B with Algorithm 2, recalculate sequence variable with Eq. (19);
7:    Update probability model by Algorithm 3;
8:    Randomly construct K swarms with decision sets P_{r,i};
9:    Allocating individuals in archives A and B to the evolutionary population by Algorithm 1;
10:   for each swarmk ∈ [1 ··· K]
11:     Update parameters of EDA and DE operations;
12:     Perform EDA and DE operations by Algorithm 4;
13:     Calculate the fitness of each individual P^t_{k,r} with Eq. (14);
14:     Perform local search Algorithm 6 for individuals by the probability of β;
15:     Update global best solutions;
16:   end for
17: until termination criterion is met
```

4. Numerical experiments

4.1. Experiments setup

Numerical experiments are tested in this paper to verify the superiority of the CHEA. All the experiments are implemented on a PC with Intel(R) Core(TM) i7-4790 CPU @3.60GHZ and 12 GB RAM. Optimal value (Opt.), the average value (Ave.), standard deviation (Std.) and computational time (CPU) are recorded down to evaluate the computational performance of algorithms.

In numerical experiments, we adopt the seven large scale instances P9-P15 from [8], which are labeled as I1-I7 in this work. The scale of these instances is listed in Table 1. And detailed data is available in [8] including unit release time, order release time, production time, and due dates. As shown in Table 1, the scale of seven instances with different timing constraints varies from 15 orders on 6 units in 2 stages to 50 orders on 20 units in 5 stages. Instances I1 ~ I4 are small and medium-size problems, and instances I5 ~ I7 are large scale problems. Instances I6 and I7 contain strict order due date timing constraints. The common parameters for CHEA are written in Table 1.

4.2. Comparison with state-of-art algorithms

4.2.1. Algorithms without considering critical path based local search strategy

To show the superiority of this work, we test the proposed algorithm on several benchmark instances without employing the critical path based local search strategy. Three evolutionary algorithms are used for comparison, including GA [42], DE [43], and cooperative evolutionary algorithm (dcEA) [40]. And the local optimization strategy is not employed for CHEA in this section, which means that line 14 in algorithm 7 is not performed for CHEA. The optimal make-span, average results, and standard deviation over 30 independent runs of experiments are listed in Table 2. It is noteworthy that, the computational costs of different evolutionary algorithms are decided by computing hardware, programming platforms, and coding skills. Hence, the computational results obtained by this paper are just for roughly understanding the efficiency of different algorithms. Hence, all the algorithms are set time limits, and all the results are obtained within time limits. CPU time limits are given in the second line. Symbol '-' means that no feasible result was found.

It is clear from Table 2 comparing to other evolutionary algorithms, CHEA performed significantly better. As problem size increased, the performance of GA, DE, and dcEA deteriorated rapidly. For instances I1 ~ I5, classic evolutionary algorithms GA and DE could get feasible solutions because of relaxed due date timing constraints. However, CC framework-based algorithms show much better performance than other evolutionary algorithms. This confirms the advantages of the CC framework for large scale MMSP. CHEA could obtain better results than dcEA for all instances. For medium-size problem I1, CHEA can reach optimal solution for every independent experiment. For instances I6 and I7, other evolutionary algorithms could not reach any feasible solution within time limits. In contrast, CHEA could get feasible solutions for two instances. Since timing constraints of due date for I6 and I7 is much tighter than other examples. Results for I6 and I7 shows great exploration ability of CHEA, for individuals could locate feasible solution space rapidly.

The superior performance of this work can be attributed to searching strategy based on the feature of MMSP. Fig. 7 shows the Gantt chart of the best solution obtained for I3 by CHEA. As depicted in Fig. 7 that, the sequence of orders in different stages follows the features that, the start time the next stage is close to the finish time in previous stage. This is a critical feature in MMSP, and the search strategy of the novel DE operation follows this feature. The sequence variables evolve based on the finish time of previous stage in non-first stages. The Gantt chart for I5 is shown in Fig. 8, which indicate the same feature. Moreover, the novel encoding method reduces the searching space for the discrete sequence variables, and it is efficient to locate the exact sequence of orders in each stage. Modified EDA operation ensures the exploration of unit assignment variables. Cooperating with DE operation, local

Table 1
Parameters of benchmark examples.

Instances	Problem size ($\|\mathbf{I}\| \times \|\mathbf{J}\| \times \|\mathbf{S}\|$)	Group SetS_K	SetS_L	f_{up}	f_{lo}	β
I1	$15 \times 6 \times 2$	{3,5,15}	{2}	0.9	0.1	0.2
I2	$10 \times 8 \times 4$	{5,10}	{2,4}			
I3	$16 \times 8 \times 2$	{4,8,16}	{2}			
I4	$30 \times 17 \times 6$	{5,15,30}	{2,4,6}			
I5	$50 \times 17 \times 6$	{5,10,25,50}	{2,4,6}			
I6	$50 \times 12 \times 6$	{5,10,25,50}	{2,4,6}			
I7	$50 \times 20 \times 5$	{5,10,25,50}	{2,3,5}			

Table 2
Results comparison of this work without local search.

Ins.	CPU(s)	GA			DE			dcEA			CHEA		
		Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
I1	300	342	367.00	15.88	275	306.20	24.08	259	280.75	19.53	235	235	0.00
I2	300	508	584.34	59.18	413	469.60	31.79	367	433.02	37.63	252	253.5	0.87
I3	300	417	484.43	46.50	275	310.95	37.99	241	257.56	12.19	210	213.95	3.20
I4	1000	30.220	34.33	2.47	30.910	33.22	2.05	25.360	26.28	0.71	20.136	20.37	0.10
I5	1000	42.870	43.08	0.11	57.870	58.15	0.75	37.010	38.70	1.00	29.443	30.07	0.44
I6	3600	-[a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	995	1061.3	64.65
I7	3600	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	545	556.65	7.86

*a. feasible solution is not obtained.

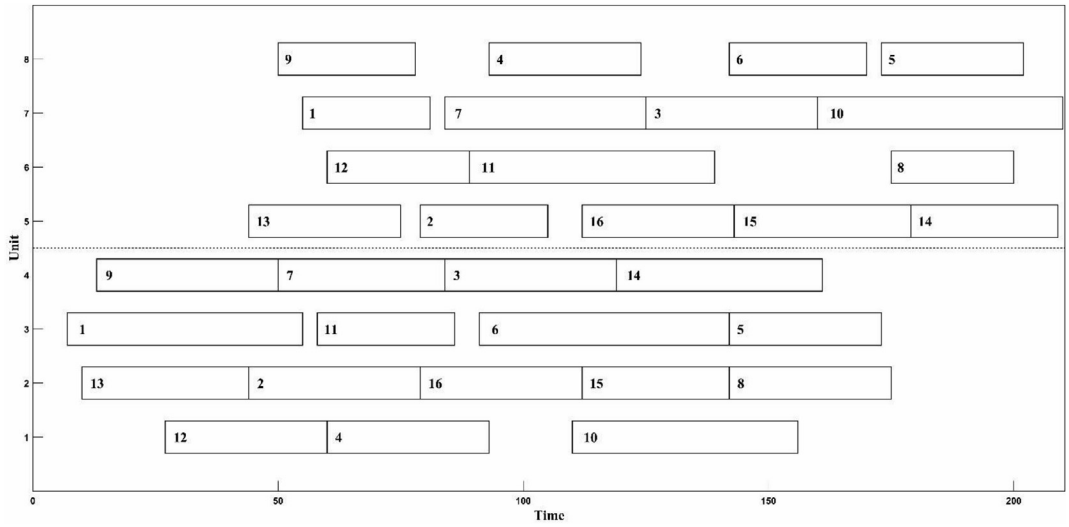


Fig. 7. Optimal solution obtained for I3.

exploitation is reinforced when $s_L < |S|$. The proposed algorithm shows better search efficiency of large scale MMSP than a single evolutionary algorithm.

4.2.2. Algorithms with local search strategy

To evaluate the influence of CHEA, statistical experiments considering local search strategy are further carried out in this section. Three evolutionary algorithms, i.e. hybrid genetic algorithm (hGA) [26], hybrid differential algorithm (hDE) [20], and cooperative evolution algorithm (dcEA) employing critical path based local search strategy, as well as a MILP-based decomposition algorithm (DA) [8] are tested for comparison in this section. All results including best solution, average value, standard deviation, and CPU time limits of 30 independent runs of experiments are written in Table 3.

As shown in Table 3, comparing to pure evolutionary algorithms, the critical path based local search strategy improves the computational performance significantly. It is obvious that, CHEA outperforms comparative evolutionary algorithms. Similar to results in Table 2, CC framework-based algorithms show a great advantage in solving medium and large size problems. For large size problem I6, dcEA/LS could reach the feasible solution space. However, it is still away from the global optimum, which proves CHEA be more adaptive than other evolutionary algorithms in solving large scale MMSP. The best solutions obtained for I6 and I7 are shown in Figs. 9 and 10 respectively. Two figures indicate the same permutation feature of MMSP that, orders' sequence is closely related to orders' finish time of the previous stage.

This work shows a slightly advantage over the MILP-based decomposition algorithm. The best solutions obtained for instances I4-I7 in 30 experiments of CHEA is better. However, as reported by average and standard deviation in Table 3, the proposed algorithm could not reach the best solution every experiment. For instance I3, only 3 optimal solutions are obtained in 30 independent experiments. It is similar for instance I4, I6, and I7, that number of better solutions obtained than decomposition method are 1, 3, and 2 respectively. This proves the efficiency of CHEA, and less

Table 3
Results comparison of this work with local search.

Ins.	CPU(s)	DA	hGA			hDE			dcEA/LS			CHEA		
			Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
I1	600	235	293	299.6	4.22	241	246	11.50	235	318.85	47.77	235	235	0.00
I2	600	252	442	468.56	31.73	268	306	48.84	266	276.1	4.54	252	252	0.00
I3	600	207	298	345.88	40.21	232	248.33	20.91	216	238.95	10.54	207	209.3	0.85
I4	1000	20.168	29.770	31.690	1.39	29.510	30.720	1.30	23.470	24.730	0.51	20.106	20.31	0.12
I5	2000	30.053	34.984	39.090	3.54	33.980	34.703	1.10	32.322	34.740	1.01	29.131	29.870	0.63
I6	5000	984	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	1080	1119.9	33.06	978	993.3	7.66
I7	5000	535	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	- [a]	534	541.3	7.67

*a. feasible solution is not obtained.

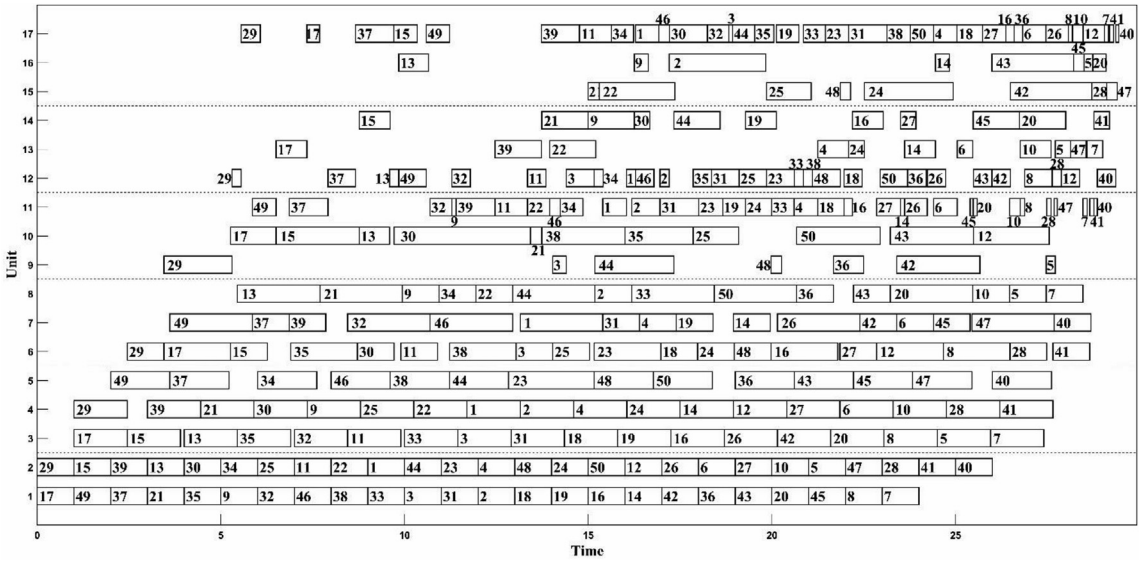


Fig. 8. Optimal solution obtained for I5.

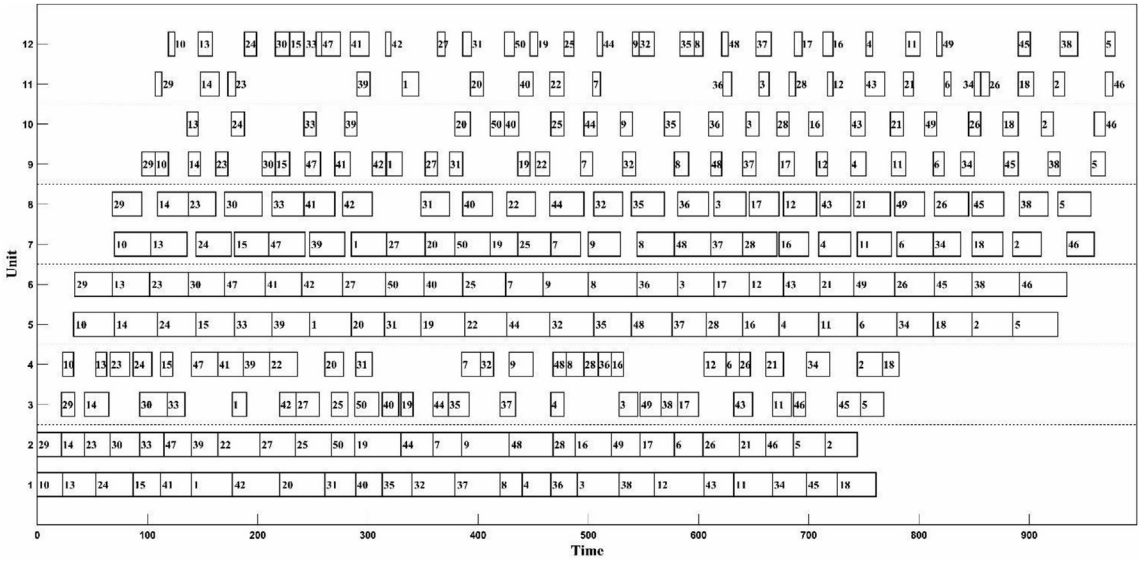


Fig. 9. Optimal solution obtained for I6.

robustness than the MILP-based decomposition method in solving large scale MMSP.

4.2.3. Convergence

To better understanding the behavior of the proposed algorithm, the evolutionary process of instance I5 is depicted in Fig. 11. For convergence performance of other instances behave similarly, only instance I5 discusses in this section. Fig. 11(a) shows the convergence rate of CHEA and comparative algorithms without employing the critical path based local search algorithm. It can be seen that CHEA not only could obtain the best results, but converge much faster than GA and DE algorithms as well. However, in the first 50 iterations, the dcEA performs better. After 200 generations, CHEA could keep searching the solution space, and other algo-

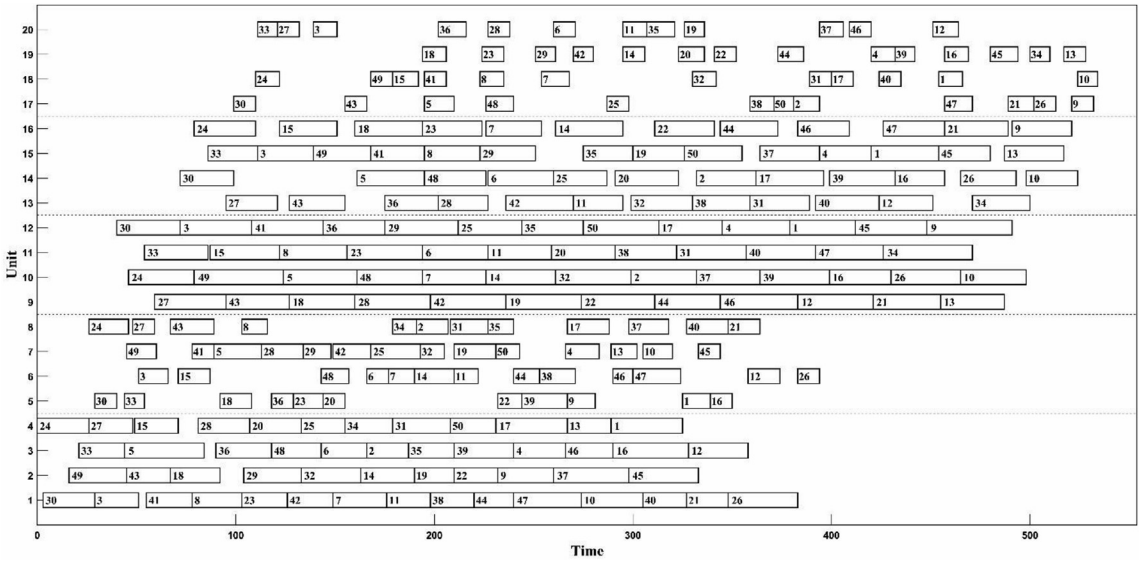
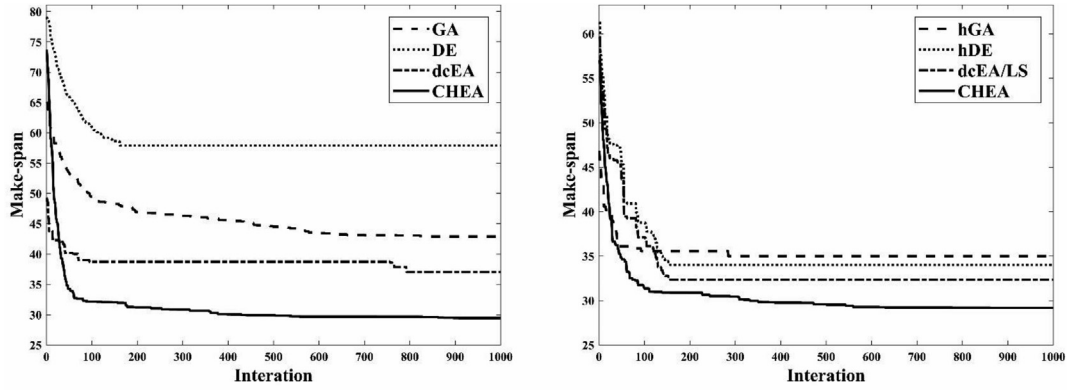


Fig. 10. Optimal solution obtained for I7.



(a) Algorithms without local search strategy (b) Algorithms with local search strategy

Fig. 11. Evolutionary process of best make-span for Instance I5.

rithms become stagnant. Fig. 11(b) shows evolutionary algorithms with the local search strategy. It is obvious that the performances of all algorithms are improved. However, this work still outperforms comparative algorithms. The effectiveness and efficiency of CHEA are especially prominent on MMSP.

4.3. Performance analysis of CHEA algorithm

In this section, we further evaluate the computational performance of the CHEA algorithm by investigating the influence of improvement in this work, including the influence of both archives, parameters, and local search strategy in CHEA. All the results are obtained based on 30 independent experiments for instances I5.

4.3.1. Influence of the archives *A* and *B*

The impact of archives **A** and **B** is first considered. Two archives are used for preserving optimal and sub-optimal solutions in each generation for CHEA. The number of archive A and B is increased from 0 to 30, and other parameters are fixed according to Table 1. The results are written in Table 4.

As shown in Table 4, the number of both archives has a significant impact on the CHEA. When $|\mathbf{A}| = 1$, the selection operation can be treated as the classic CC framework based evolutionary algorithm. While considering a single archive **A** or **B**, CHEA could not obtain better results than that with both archives in most cases. If only archive **B** is applied, i.e. $|\mathbf{A}| = 0$, the possibility of trapping into local optimization might be increased. This can be proved by the first row in Table 11, and best solutions are much worse than other results. The results with both archives show much better performance. Since the diversity of the population can be improved by selected individuals with different strategies. However, the number of both archives is not simply the larger the better. The performance of CHEA seems to be less effective as both archives number increase as shown in the last row and last column. The results show that the number of archives set to $|\mathbf{A}| = 5$, $|\mathbf{B}| = 10$ appears to an ideal choice for CHEA.

4.3.2. Influence of the parameter s_L and group size

Parameter s_L and group size K are closely related to the encoding method in this work. So the influence of both parameters is evaluated in this section. The group set S_K is set {2, 4, 6} for instance I5, and group size varies from 5 to 50. The influence of parameters s_L and group size is reported in Table 5 for instance I5.

Results show that, two parameters have a great impact on the computational performance of CHEA. It is obvious that, as group size K increases, the searching efficiency is improved significantly.

Table 4
Influence of number of archive A and B.

|A|	|B|=0			|B|=5			|B|=10			|B|=20			|B|=30		
	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
0	–	–	–	32.061	32.127	0.043	32.317	32.517	0.156	31.757	32.233	0.308	32.21	32.639	0.265
1	30.637	31.391	0.545	–	–	–	–	–	–	–	–	–	–	–	–
5	30.302	30.946	0.447	29.735	30.057	0.253	29.131	29.870	0.530	29.433	29.939	0.367	29.712	30.111	0.298
10	30.286	30.961	0.453	30.259	31.139	0.709	30.184	30.762	0.547	29.923	30.620	0.456	30.244	31.211	0.400
20	30.297	30.997	0.398	30.607	31.546	0.603	30.313	31.099	0.564	30.612	31.418	0.608	30.531	31.458	0.481
30	30.068	31.044	0.456	29.94	31.465	1.012	30.688	31.222	0.432	30.814	31.596	0.707	31.062	31.735	0.693

Table 5
Influence of parameters s_L and group size.

s_L	S_K={5,10,25,50}			S_K={5,10}			S_K={10,25}			S_K={25,50}		
	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
{2}	72.897	77.055	10.761	71.185	86.581	16.45	79.555	120.51	59.726	79.161	88.2923	22.128
{4}	45.32	46.358	2.060	54.898	59.313	2.324	51.635	55.8386	4.624	48.561	50.4613	3.748
{6}	30.126	30.436	0.236	35.781	38.942	2.228	35.781	38.1598	1.952	30.982	31.5272	0.467
{2,4}	56.063	58.396	4.870	61.336	65.273	1.669	60.073	63.4371	3.443	60.049	62.1104	3.625
{4,6}	30.006	30.334	0.210	36.613	39.005	1.395	34.693	35.7447	0.764	30.977	31.8217	0.364
{2,4,6}	29.131	29.870	0.530	36.696	40.178	2.331	35.931	36.7938	0.574	30.963	31.428	0.283

The optimal group set is {5,10,25,50}, which makes the variables in each sub-individual vary from 10 to 1. Since smaller group size K increase the exploration, and larger group size K increases exploitation ability. Multiple group size can balance exploitation and exploration effectively. Novel encoding scheme in this paper is based on different jobs instead of operations. So when group size equals 1, i.e. $K = 1$, there are still 6 unit assignment and sequence variables of a single order in different stages that participate in the evolutionary operation.

For the parameter s_L, greater value means better exploration and less exploitation as discussed in section 3.4.2. When set $s_L = 6$, the proposed algorithm could obtain a promising solution to the scheduling problem. Since $s_L = 6$ enhance the exploration ability of CHEA, which can be seen in third row. Smaller value of s_L improves local exploitation, and limits the global searching efficiency. Similar to group size, both parameters with multiple s_L could reach a balance between exploration and exploitation. Here it is noteworthy that, when $s_L < 6$, and $K = 50$, lots of offspring with similar solutions might be generated. This have a great impact on the diversity of CHEA. So different selection methods and modified EDA operation are proposed to handle this problem.

4.3.3. Influence of learning rate LR and degenerating rate DR

Experiments are carried out in Table 6 to observe the influence of learning rate LR and degenerating rate DR. Learning rate LR varies from 0.01 to 0.30, and DR from 0.00 to 0.20. From Table 6, the impact of DR on CHEA is not obvious. Since many settings of LR and DR could reach a promising performance of proposed work. The ideal choice for two parameters seems to be $LR = 0.10$, $DR = 0.10$ in this work.

The results in Table 6 provide an effective choice for LR and DR in solving MMSP. However, it is not sufficient to show the effectiveness of parameter DR, which is used to prevent the premature of probability vector in EDA operation. As discussed in Section 3.4.1, DR could set upper and lower bounds for probability vectors. Hence, we recorded the probability vector of order 2 in stage 1 to indicate the effect of DR. As shown in Table 7, the probability vector of $p^t_{i2,j1}$ and $p^t_{i2,j2}$ with different settings of LR, DR, and iteration number are listed. It is can be seen from results that, while $DR = 0$ and $iteration > 100$, the probability vectors for unit assignment vector easily trap into local optimization, i.e. $p^t_{i2,j1} = 0$. The smaller value of DR/LR shows a similar performance. For example, while $LR = 0.2$ and $DR = 0.01$, the probability vectors keep unchanged for 500 generations. This indicates that, individuals are trapped into local optimization, and the exploration ability of CHEA is limited. As DR increase with iteration, the influence of LR is gradually decreased. While $DR = 0.20$, the variation of probability is slight, which strengthens exploration and weakens the exploitation. This example is very representative in solving MMSP for CHEA. Since probability vectors for other orders follow similar trends. As analyzed in the previous section, while $s_L < 6$, not all the variables in the unit assignment and order sequence vectors participate in evolving operation. Numbers of similar offsprings have a great impact on the probability model. It is worth mentioning that, modified EDA operation is only introduced based on the proposed algorithm in this work, which does not apply to other algorithms.

4.3.4. Influence of critical path based local search strategy

As discussed above, CHEA shows good exploration and exploitation ability. For DE operation shares part of local searching ability

Table 6
Influence of parameters LR and DR.

LR	DR = 0.00			DR = 0.01			DR = 0.05			DR = 0.10			DR = 0.20		
	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
0.01	30.396	30.706	0.303	30.027	30.384	0.307	30.061	30.424	0.491	29.954	30.336	0.414	30.246	30.622	0.419
0.05	29.986	30.738	0.798	30.087	30.415	0.333	30.393	30.693	0.391	30.378	30.662	0.234	30.154	30.606	0.493
0.10	31.029	31.399	0.389	30.183	30.501	0.253	29.894	30.332	0.309	29.131	29.870	0.530	29.802	30.211	0.365
0.20	30.556	31.115	0.416	30.436	30.693	0.218	29.823	30.257	0.295	30.203	30.591	0.241	30.35	30.672	0.213
0.30	30.637	31.457	0.598	30.072	30.765	0.328	30.164	30.644	0.315	29.716	30.472	0.415	30.383	30.804	0.229

Table 7
Probabilistic vector of $p^t_{i2,j1}$ and $p^t_{i2,j2}$ for instance I5.

Iteration	LR	DR = 0		DR = 0.01		DR = 0.05		DR = 0.10		DR = 0.20	
		$p^t_{i2,j1}$	$p^t_{i2,j2}$	$p^t_{i2,j1}$	$p^t_{i2,j2}$	$p^t_{i2,j1}$	$p^t_{i2,j2}$	$p^t_{i2,j1}$	$p^t_{i2,j2}$	$p^t_{i2,j1}$	$p^t_{i2,j2}$
***t* = 100**	**0.01**	0.185	0.815	0.716	0.284	0.513	0.487	0.508	0.492	0.524	0.476
	0.05	0.997	0.003	0.344	0.656	0.750	0.250	0.650	0.350	0.550	0.450
	0.1	0.750	0.250	0.256	0.955	0.751	0.249	0.611	0.389	0.667	0.333
	0.2	1.000	0.000	0.960	0.040	0.100	0.900	0.833	0.167	0.650	0.350
	0.3	1.000	0.000	0.740	0.260	0.183	0.817	0.482	0.518	0.580	0.420
***t* = 500**	**0.01**	0.003	0.997	0.750	0.250	0.583	0.417	0.545	0.455	0.500	0.500
	0.05	1.000	0.000	0.123	0.877	0.750	0.250	0.667	0.333	0.480	0.520
	0.1	0.761	0.239	0.115	0.885	0.833	0.167	0.700	0.300	0.667	0.333
	0.2	1.000	0.000	0.976	0.024	0.100	0.900	0.833	0.167	0.551	0.449
	0.3	1.000	0.000	0.984	0.016	0.071	0.929	0.475	0.525	0.470	0.530
***t* = 1000**	**0.01**	0.000	1.000	0.750	0.250	0.550	0.450	0.545	0.455	0.495	0.505
	0.05	1.000	0.000	0.083	0.917	0.350	0.650	0.667	0.333	0.480	0.520
	0.1	0.800	0.200	0.046	0.954	0.833	0.167	0.650	0.350	0.467	0.533
	0.2	1.000	0.000	0.976	0.024	0.100	0.900	0.833	0.167	0.450	0.550
	0.3	1.000	0.000	0.984	0.016	0.071	0.929	0.491	0.509	0.440	0.560

in this algorithm, while $s_L < |S|$. Cooperating with the critical path-based local search algorithm, CHEA shows better computational performance. To investigate the effectiveness of critical path based local search strategy, the results of dcEA/LS, $CHEA_1$, and CHEA, within time limits are listed in Table 8. Here, $CHEA_1$ is the proposed algorithm without considering the critical path based local search strategy.

As shown in Table 8, critical path based local search strategy can further improve the local optimization solution for CHEA. However, the impact is not as obvious as other evolutionary algorithms. Without employing local search strategy, $CHEA_1$ also shows good exploration and exploitation ability. Modified EDA and DE operations enhance global search ability. And when $s_L < |S|$, DE operation share part of local search ability. It worth mentioning that, critical path based local search strategy is designed for FJSP. Although similar to FJSP, MMSP still has specific features, which might influence the search efficiency of critical path based local search strategy. In the contrary, The evolutionary operations of CHEA is proposed based on the feature of MMSP, which highly improves the search efficiency.

In Table 9, the average computational time of each iteration for evolutionary algorithms are listed. dcEA/LS employs the distributed parallel process units in original work, which is not applied in this paper. As results showed, the computing time of CHEA is slightly higher than in GA and DE. The computational cost of evolutionary algorithms with local search strategy increases sharply about 40%-50% over pure algorithms. We can see that CHEA performs much better than comparison algorithms without increase much CPU time. This also indicates the superiority of this work that, and proposed evolutionary operations based on the feature of MMSP is efficient in solving this problem.

5. Conclusion

In this paper, CHEA based on the CC algorithm is proposed for solving large scale MMSP. Considering the feature of MMSP, a new encoding method is proposed by grouping the operation in all stages of one order into one decision set. Based on this encoding scheme, the set-based random grouping strategy is adopted, and

Table 8
Influence of critical path based local search.

Ins.	CPU(s)	dcEA/LS			$CHEA_1$			CHEA		
		Opt.	Avg.	Sd.	Opt.	Avg.	Sd.	Opt.	Avg.	Sd.
I1	600	235	318.85	47.77	235	235	0.00	235	235	0.00
I2	600	266	276.1	4.54	252	253.10	0.79	252	252	0.00
I3	600	216	238.95	10.54	210	213.85	3.11	207	209.3	0.85
I4	1000	23.470	24.730	0.51	20.136	20.320	0.95	20.106	20.310	0.12
I5	2000	32.322	34.740	1.01	29.443	29.880	0.32	29.131	29.870	0.63
I6	5000	1080	1119.9	33.06	995	1061.3	64.65	978	993.3	7.66
I7	5000	- [a]	- [a]	- [a]	545	556.65	7.86	534	541.3	7.67

*a. feasible solution is not obtained.

Table 9
Average computational time of each iteration.

Ins.	CPU time(s)							
	GA	hGA	DE	hDE	dcEA	dcEA/LS	$CHEA_1$	CHEA
I1	0.31	0.50	0.26	0.45	0.40	0.80	0.30	0.61
I2	0.32	0.51	0.27	0.52	0.44	0.95	0.33	0.60
I3	0.41	0.79	0.42	0.80	0.59	1.07	0.45	0.82
I4	0.49	0.82	0.45	0.85	0.61	1.10	0.51	0.89
I5	1.01	2.02	1.05	2.06	1.32	2.51	1.12	2.03
I6	1.30	2.22	1.33	2.33	1.50	2.89	1.20	2.39
I7	1.34	2.35	1.40	2.37	1.59	3.07	1.27	2.43

the population can be decomposed into sub-populations in the CC framework. To improve the diversity of population, two extra archives that represent different selection strategies are proposed. Through novel assignment and selection operations, individuals of each swam can evolve based on multiple best solutions in each generation. Evolutionary operations are also modified to fit in the novel encoding method. First, a parameter DR is introduced for the PBIL method that widely used in EDA to prevent the algorithm trapped into local optimization. A novel DE operation with multiple evolving strategies is designed to adjust the feature of MMSP. And the exploitation and exploration can be greatly balanced by both novel operation. To further enhance the local optimization, the critical path based local searching algorithm is embedded into the hybrid algorithm. Several instances are constructed to verify the superiority of this work. Proposed CHEA shows better search efficiency in solving large scale MMSP than comparative evolutionary algorithms. Moreover, the influence of improvement in CHEA is also investigated in this work. All the results show that, proposed CHEA shows good ability of exploration and exploitation in dealing with large scale MMSP.

Although CHEA shows great advantage in solving large scaling MMSP, other actual production batch plant scheduling constraints are not taken into consideration in this work. In the future, our work will focus on more in-depth analysis of MMSP with more real-world characteristics. For example, scheduling constraints with different inter-storage policies and optimization objectives also need to be further studied. Moreover, other evolutionary algorithms could be employed as the subcomponent optimizer within the CC framework. It will be expected to find more efficient optimization methods on this problem.

CRediT authorship contribution statement

Yuxin Han: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft. **Xingsheng Gu:** Project administration, Funding acquisition, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was financially sponsored by the National Science Foundation of China (Grant No. 61973120, 61573144, 61773165, 61673175), the Program of Introducing Talents of Discipline to Universities (the 111 Project) (Grant No. B17017), Fundamental Research Funds for the Central Universities (No. 222201917006).

References

[1] C.A. Méndez, J. Cerdá, I.E. Grossmann, I. Harjunkoski, M. Fahl, State-of-the-art review of optimization methods for short-term scheduling of batch processes, Comput. Chem. Eng. 30 (6-7) (2006) 913–946.
[2] F.M. Novara, J.M. Novas, G.P. Henning, A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation, Comput. Chem. Eng. 93 (2016) 101–117.
[3] S. Velez, C.T. Maravelias, Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling, Comput. Chem. Eng. 53 (2013) 70–85.
[4] J.M. Pinto, I.E. Grossmann, A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants, Ind. Eng. Chem. Res. 34 (9) (1995) 3037–3051.
[5] J.M. Pinto, I.E. Grossmann, An alternate MILP model for short-term scheduling of batch plants with preordering constraints, Ind. Eng. Chem. Res. 35 (1) (1996) 338–342.
[6] S. Gupta, I.A. Karimi, An improved MILP formulation for scheduling multiproduct, multistage batch plants, Ind. Eng. Chem. Res. 42 (11) (2003) 2365–2380.
[7] P.A. Marchetti, J. Cerdá, An approximate mathematical framework for resource-constrained multistage batch scheduling, Chem. Eng. Sci. 64 (11) (2009) 2733–2748.
[8] P.M. Castro, I. Harjunkoski, I.E. Grossmann, Optimal short-term scheduling of large-scale multistage batch plants, Ind. Eng. Chem. Res. 48 (24) (2009) 11002–11016.
[9] P.M. Castro, I. Harjunkoski, I.E. Grossmann, Greedy algorithm for scheduling batch plants with sequence-dependent changeovers, AIChE J. 57 (2) (2011) 373–387.
[10] P.M. Castro, I.E. Grossmann, A.Q. Novais, Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers, Ind. Eng. Chem. Res. 45 (18) (2006) 6210–6226.
[11] L.J. Zeballos, J.M. Novas, G.P. Henning, A CP formulation for scheduling multiproduct multistage batch plants, Comput. Chem. Eng. 35 (12) (2011) 2973–2989.
[12] F.M. Novara, J.M. Novas, G.P. Henning, A comprehensive CP approach for the scheduling of resource-constrained multiproduct multistage batch plants Comput. Aided Chem. Eng. 32 (2013) 589–594, https://doi.org/:10.1016/B978-0-444-63234-0.50099-3.
[13] Y. He, C.-W. Hui, Genetic algorithm for large-size multi-stage batch plant scheduling, Chem. Eng. Sci. 62 (5) (2007) 1504–1523.
[14] B. Shi, L.-X. Yan, W. Wu, Rule-based scheduling of single-stage multiproduct batch plants with parallel units, Ind. Eng. Chem. Res. 51 (25) (2012) 8535–8549.
[15] B. Shi, X. Qian, S. Sun, L. Yan, Rule-based scheduling of multi-stage multi-product batch plants with parallel units, Chin. J. Chem. Eng. 25 (8) (2017) 1022–1036.
[16] I. Ali, A. Ali, A research survey : review of flexible job shop scheduling techniques, Intl. Trans. in Op. Res. 23 (2016) 551–591, https://doi.org/10.1111/itor.12199.
[17] L. Lin, M. Gen, Hybrid evolutionary optimization with learning for production scheduling: state-of-the-art survey on algorithms and applications, Int. J. Prod. Res 56 (2018) 193–223, https://doi.org/10.1080/00207543.2018.1437288.
[18] G. Vilcot, J.-C. Billaut, A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem, Int. J. Prod. Res. 49 (23) (2011) 6963–6980.
[19] H. Na, J. Park, Multi-level job scheduling in a flexible job shop environment, Int. J. Prod. Res. 52 (13) (2014) 3877–3887.
[20] Y. Yuan, H. Xu, Flexible job shop scheduling using hybrid differential evolution algorithms, Comput. Ind. Eng. 65 (2) (2013) 246–260.
[21] X. Shao, W. Liu, Q. Liu, C. Zhang, Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem, Int. J. Adv. Manuf. Technol. 67 (9-12) (2013) 2885–2901.
[22] K.Z. Gao, P.N. Suganthan, Q.K. Pan, T.J. Chua, T.X. Cai, C.S. Chong, Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives, J. Intell. Manuf. 27 (2) (2016) 363–374.
[23] L. Wang, S. Wang, Y.e. 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.
[24] R. Pérez-Rodríguez, A. Hernández-Aguirre, A hybrid estimation of distribution algorithm for flexible job-shop scheduling problems with process plan flexibility, Appl. Intell., 48 (2018) 3707–3734. https://doi.org/0.1007/s10489-018-1160-z.
[25] M. Amiri, M. Zandieh, M. Yazdani, A. Bagheri, A variable neighborhood search algorithm for the flexible job-shop scheduling problem, Int. J. Prod. Res. 7543 (2010), https://doi.org/10.1080/00207540903055743.
[26] J. Gao, L. Sun, M. Gen, A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Comput. Oper. Res. 35 (9) (2008) 2892–2907.
[27] O. Sobeyko, L. Mönch, Heuristic approaches for scheduling jobs in large-scale flexible job shops, Comput. Oper. Res. 68 (2016) 97–109.
[28] E.S. Nicoara, F.G. Filip, N. Paraschiv, Simulation-based optimization using genetic algorithms for multi-objective flexible JSSP, Int. J. Prod. Res 20 (2011) 333–344.
[29] J. Xiong, X.u. Tan, K.-W. Yang, L.-N. Xing, Y.-w. Chen, A hybrid multiobjective evolutionary approach for flexible job-shop scheduling problems, Math. Probl. Eng. 2012 (2012) 1–27.
[30] P.-H. Lu, M.-C. Wu, H. Tan, Y.-H. Peng, C.-F. Chen, A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems, J. Intell. Manuf. 29 (1) (2018) 19–34.
[31] L. De Giovanni, F. Pezzella, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, Eur. J. Oper. Res. 200 (2) (2010) 395–408.
[32] M.A. Potter, K.A. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.-.-P. Schwefel, R. Männer (Eds.), Parallel Probl. Solving from Nat. –– PPSN III, Springer, Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 249–257, https://doi.org/10.1007/3-540-58484-6_269.
[33] F. vandenBergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Computat. 8 (3) (2004) 225–239.

[34] Z. Yang, K.e. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999.
[35] M.N. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, IEEE Congr. Evol. Comput. (CEC), IEEE 2010 (2010) 1–8, https://doi.org/10.1109/CEC.2010.5586127.
[36] M.N. Omidvar, X. Li, S. Member, Cooperative Co-evolution with Delta Grouping for Large Scale Non-separable Function Optimization, in: 2010 IEEE Congr. Evol. Comput. (CEC), IEEE, 2010: pp. 1–8. https://doi.org/10.1109/CEC.2010.5585979.
[37] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput. 18 (2014) 378–393. https://doi.org/10.1109/TEVC.2013.2281543.
[38] X. Li, S. Member, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2012) 210–224, https://doi.org/10.1109/TEVC.2011.2112662.
[39] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: R. Schaefer, C. Cotta, J. Kołodziej, G. Rudolph (Eds.), Parallel Probl. Solving from Nature, PPSN XI, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 300–309. https://doi.org/10.1007/978-3-642-15871-1_31.
[40] Lu Sun, Lin Lin, Haojie Li, Mitsuo Gen, Large scale flexible scheduling optimization by a distributed evolutionary algorithm, Comput. Ind. Eng. 128 (2019) 894–904.
[41] Ming-Gang Dong, Ning Wang, A novel hybrid differential evolution approach to scheduling of large-scale zero-wait batch processes with setup times, Comput. Chem. Eng. 45 (2012) 72–83.
[42] H. Chen, J. Ihlow, C. Lehmann, A genetic algorithm for flexible job-shop scheduling, IEEE Int. Conf. Robot. Autom., IEEE 1999 (1999) 1120–1125.
[43] K. Price, R.M. Storn, J.A. Lampinen, Differential evolution: A practical approach to global optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. https://doi.org/10.1007/3-540-31306-0.

Yuxin Han was born in 1990. He received the B.E. degree in automation from East China University of Science and Technology in 2013. He is pursuing the Ph.D. degree in Department of Automation, ECUST. His main search area includes intelligent optimization and scheduling methods of batch plants.

Xingsheng Gu received the B.S. degree from Nanjing Institute of Chemical Technology in 1982, M.S. and Ph. D. degree from East China University of Chemical Technology in 1988 and 1993, respectively. He is currently a professor at School of Information Science and Engineering, East China University of Science and Technology. His research interests include planning and scheduling for process industry, modeling, control and optimization for industry processes, intelligent optimization, faults detection and diagnosis, etc.