Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.040215

Article

Competitive and Cooperative-Based Strength Pareto Evolutionary Algorithm for Green Distributed Heterogeneous Flow Shop Scheduling

Kuihua Huang[1], Rui Li[2], Wenyin Gong[2,*], Weiwei Bian[3] and Rui Wang[1]

[1]College of System Engineering, National University of Defense Technology, Changsha, 410073, China
[2]School of Computer Science, China University of Geosciences, Wuhan, 430074, China
[3]Equipment General Technology Laboratory, Beijing Mechanical Equipment Research Institute, Beijing, 100854, China

*Corresponding Author: Wenyin Gong. Email: wygong@cug.edu.cn

Received: 09 March 2023; Accepted: 27 April 2023; Published: 23 June 2023

Abstract: This work aims to resolve the distributed heterogeneous permutation flow shop scheduling problem (DHPFSP) with minimizing makespan and total energy consumption (TEC). To solve this NP-hard problem, this work proposed a competitive and cooperative-based strength Pareto evolutionary algorithm (CCSPEA) which contains the following features: 1) An initialization based on three heuristic rules is developed to generate a population with great diversity and convergence. 2) A comprehensive metric combining convergence and diversity metrics is used to better represent the heuristic information of a solution. 3) A competitive selection is designed which divides the population into a winner and a loser swarms based on the comprehensive metric. 4) A cooperative evolutionary schema is proposed for winner and loser swarms to accelerate the convergence of global search. 5) Five local search strategies based on problem knowledge are designed to improve convergence. 6) A problem-based energy-saving strategy is presented to reduce TEC. Finally, to evaluate the performance of CCSPEA, it is compared to four state-of-art and run on 22 instances based on the Taillard benchmark. The numerical experiment results demonstrate that 1) the proposed comprehensive metric can efficiently represent the heuristic information of each solution to help the later step divide the population. 2) The global search based on the competitive and cooperative schema can accelerate loser solutions convergence and further improve the winner's exploration. 3) The problem-based initialization, local search, and energy-saving strategies can efficiently reduce the makespan and TEC. 4) The proposed CCSPEA is superior to the state-of-art for solving DHPFSP.

Keywords: Distributed heterogeneous flow shop scheduling; green scheduling; SPEA2; competitive and cooperative

1 Introduction

With the development of international trade and the global economy, enterprises receive more and more orders. However, the traditional centralized manufacturing model cannot meet the fast-producing requirement of the market [1]. Because to occupy more parts of the market, the enterprises reduce the production due date in succession but the total processing time cannot be changed. Thus, the enterprise will open multiple factories to finish orders and this mode is called distributed manufacturing. Flow shop scheduling problems [2] are one of the most classical combinational optimization problems which have been deeply researched. As its extension problem, the distributed heterogeneous permutation flow shop scheduling problem (DHPFSP) has attracted more and more attention in recent years [3]. In DHPFSP, the processing time, machine number, machine type, worker number or *et al.* is different which makes DHPFSP has factory flexibility. Meanwhile, the DHPFSP is harder to solve than traditional distributed identical permutation flow shop scheduling problems. Thus, studying DHPFSP can provide theoretical guidance to practical manufacturing. With the growing global temperature, green scheduling gets more and more attention [4]. Moreover, total energy consumption (TEC) is the critical indicator that reflects carbon emissions. Thus, to ensure sustainable and green manufacturing, this work aims to minimize makespan and TEC in DHPFSP.

The main methods for DHPFSP include the iterative greedy algorithm [5], memetic algorithm [6], cooperative algorithm [7], Pareto-based genetic algorithm [8], and decomposition-based algorithm [9]. Moreover, Huang et al. [3] proposed a bi-roles co-evolutionary (BRCE) framework which can greatly balance computation resources between global and local searches. However, BRCE and previous works are hard to explore during the global search for the following reasons: i) They ignore the heuristic information of each solution. ii) The fast no-dominated sorting environment selection strategy can not balance convergence and diversity well.

Strength Pareto evolutionary algorithm (SPEA2) gets many concerns in multi-objective optimization problems due to its special fitness representation and good performance [10]. Thus, adopting SPEA2 for DHPFSP can balance convergence and diversity. Competitive swarm optimization (CSO) is also popular because CSO uses the heuristic information of each solution and lets the loser swarm learn from the winner swarm which accelerates the converging of global search [11]. So combining CSO and SPEA2 can improve the convergence of global search which is never proposed for DHPFSP before.

This study aims to resolve the green DHPFSP (GDHPFSP) by minimizing makespan and TEC. To solve GDHPFSP, a competitive and cooperative strength Pareto evolutionary algorithm (CCSPEA) is proposed for GDHPFSP. **The main contributions are summarized as follows**: 1) An improved fitness function from SPEA2 is used to represent the heuristic information of each solution. By combining the convergence metric and diversity metric, the comprehensive performance of each solution can be represented. 2) A competitive and cooperative genetic operator is proposed to rapidly converge. According to the heuristic information, the population can be divided into winner and loser swarms. Then, a cooperative genetic operator is designed for each swarm to accelerate converging. Finally, a separation experiment is designed on 22 DHPFSP instances. The results demonstrate the effectiveness of each improvement. Moreover, CCSPEA is compared to five state-of-arts and the results state the superiority of CCSPEA.

The rest of this manuscript is organized as follows: The literature review is introduced in Section 2. The description of DHPFSP are introduced in Section 3. Section 4 illustrates the proposed CCSPEA. The detailed results of the experiments are explained and discussed in Section 5. Finally, Section 6 concludes this work and discusses future directions.

2 Literature Review

2.1 Related Work of SPEA2

SPEA2 [10] has attracted much attention for multi-objective optimization problems due to its convergence speed and performance. Sahoo et al. [12] combined SPEA2 and particle swarm optimizer for electrical distribution systems optimization problems. Li et al. [13] applied the deep-q-learning algorithm to improve the performance of shift-based density estimation SPEA2 (SPEA2 + SDE). Xu et al. [14,15] applied SPEA2 in edge computing to optimize the Internet of Things. Liu et al. [16] adopted SPEA2 + SDE for the STATCOM allocation problem and got good results. Biswas et al. [17,18] adopted SPEA2 to charge transportation problems and get better performance than state-of-arts. Maurya et al. [19] used SPEA2 to smart home appliances scheduling problems and got the best results. Luo et al. [20] improved SPEA2 to find top-k solutions in preference-based multi-objective optimization problems and obtained great performance. Cao et al. [21] designed an improved SPEA2 based on the relationship propagation chain for iron-steel scheduling. Amin-Tahmasbi used to apply SPEA2 to solve flow shop problems and got good results [22]. Based on the introductions above, it is obvious that SPEA2 is an effective algorithm for the discrete multi-objective optimization problem.

2.2 Related Work of CSO

Instead of the traditional partial swarm optimizer (PSO) algorithm, a novel pairwise competitive schema-based swarm optimizer was proposed, which is called the competitive swarm optimizer (CSO) [11]. Unlike the classical PSO [23,24], CSO divides the population of each generation into a winner group and a loser group. Then, the winner only executes mutation to improve itself and the loser learns critical knowledge from the selected winner to generate a new solution.

$$\begin{cases} v_l(t+1) = r_0 v_l(t) + r_1(x_w(t)' - x_l(t)'), \\ x_l(t+1) = x_l(t) + v_l(t+1), \\ x_w(t)' = x_w(t) + r_0 v_w(t), \\ x_l(t)' = x_l(t) + r_0 v_l(t), \end{cases} \tag{1}$$

The CSO is proposed for the continuous optimization problem and the main learning schema is vector difference which is shown in Eq. (1). Due to its fast-learning feature, the CSO is usually used to solve multi/many-objective optimization problems (MOP) [25,26]. Gu et al. improved the initialization and learning strategies of CSO which can get better balance convergence and diversity [27]. Huang et al. considered parameter adaptive CSO to improve its intelligence [28]. Large-scale MOP is the main research field of CSO [29] and many improved CSO has been proposed. Mohapatra et al. designed a tri-competitive schema-based CSO to improve exploration [30–32]. Ge et al. proposed inverse modeling to update the winners to accelerate the convergence of CSO [33]. Liu et al. designed three different competitive schemas to improve the diversity of CSO [34]. Qi et al. designed the neighborhood search strategy to enhance CSO [35]. Chen et al. divided CSO into three phases and got better results than state-of-arts. Moreover, CSO also can be applied to constrained MOP [36], many-objective optimization problems [37], feature selection [38], and wireless sensor networks [39].

2.3 Related Work of DHPFSP

DHPFSP has attracted more and more attention in recent years. Chen et al. solved DHPFSP with the content machine speed by an improved estimation distribution algorithm which got better performance than state-of-arts [6]. Shao et al. designed several strategies of local search for DHPFSP to improve the convergence of the algorithm [40]. To balance the convergence and diversity, Li et al. improved the MOEA/D with the behavior of bee swarm for DHPFSP which enhances the convergence [9]. Zhao et al. developed a self-learning framework for operator selection that enhances the convergence of local search [41]. Mao et al. proposed the hash map to store the candidate solution which improves the ability of global search for DHPFSP [8]. Meng et al. considered DHPFSP with lot-streaming and sequence-dependent set-up time and proposed an algorithm of artificial bee colony to solve it [42]. Lu et al. combined hybrid flow shop and flow shop as DHPFSP [43]. Huang et al. proposed the BRCE algorithm for DHPFSP which divided global and local search into two populations and balanced convergence and diversity [3].

2.4 Research Gap and Motivation

The previous algorithm BRCE [3] gets the best performance on DHPFSP. However, there are some disadvantages to BRCE which are stated as follows: 1) The BRCE randomly selects a parent from the mating pool to generate offspring but does not use the heuristic information of population. However, CSO [11] can divide the population into a winner swarm and a loser swarm which can enhance global search by using the heuristic information of the population. 2) The BRCE adopts environment selection by fast non-dominated sorting which keeps convergence first and diversity later. This strategy always focuses on the last front of all fronts but considers little about crowded solutions in the former fronts. However, SPEA2 [10] can measure convergence and diversity by a comprehensive metric that can better balance the convergence and diversity of the whole population. Thus, SPEA2 is applied to better represent the heuristic information of each solution. 3) The BRCE executes crossover by particle match crossover (PMX) [3] to generate offspring. However, the decision space of DHPFSP is so large and PMX is too weak to sufficiently search the potential solutions.

Thus, based on the discussion above, this work proposed a competitive and cooperative strength Pareto evolutionary algorithm (CCSPEA) for DHPFSP to overcome the disadvantages of BRCE.

3 Problem Statement

In DHPFSP, there are total n_f heterogeneous factories and n jobs. Each job needs to be processed by m stage in each factory and each stage has only one machine to process. Each machine $M_{f,k}$ can adjust processing speed $v_{f,i,k}$ from $\{v_1, \ldots, v_5\}$ when processing job I_i. The every job's original processing time on every stage in each factory is $p^o{}_{f,i,j}$ and the real processing time is $p^o{}_{f,i,j}/v_{f,i,k}$. Meanwhile, $p^o{}_{f,i,j}$ is different at the same stage in each factory. There are three sub-problems in DHPFSP: i) determine the factory assignment of each job; ii) ensure the processing sequence of all jobs in every heterogeneous factory; and iii) assign a processing speed for each job on each stage of the machine. The objectives are to minimize makespan and TEC. It is worth noting that the machine work power is $p^r{}_{f,i,j} * v_{f,i,k}{}^2$.

DHPFSP's assumptions of in this work are described in following parts: i) At time zero, every job starts being processed at stage one. ii) All machines are not allowed to process other jobs when it is processing one job. iii) Each job can only be assigned to single factory. Meanwhile, every job is not permitted to be processed on two different machines at the same time. iv) Every machine's energy consumption of and processing times of each job is certain. v) Transportation and setup constraints are not studied. Furthermore, dynamic events are not studied. vi) Every job's processing time of on the same stage in different heterogeneous factories is different.

The MILP model is the same as reference [3]. Thus, the notations and model are not introduced in this work.

4 Our Approach: CCSPEA

4.1 Framework of CCSPEA

Algorithm 1 states the framework of CCSPEA. First, CCSPEA is an improvement of BRCE [3], and CCSPEA also divides global and local searches into two swarms P and C. Meanwhile, P is initialized by three heuristic rules to permit great convergence and diversity. Second, P will be divided into winner W and loser L by competitive strategy. Next, the cooperative evolution is adopted to let L learn from W and accelerate converging. Then, W executes self-evolution to search for potential non-dominated solutions. Moreover, after the environmental selection, multiple neighborhood structure search and energy-saving strategies are adopted to enhance convergence. In the end, the ultimate non-dominated solutions set is got from consumer swarm C.

4.2 Encoding and Decoding

Encoding schema: In proposed approach CCSPEA, job sequence (*JS*), factory assignment (*FA*), and speed selection (*SS*) are represented by two vectors and a matrix. Fig. 1 shows the encoding schema for DHPFSP. In *FA* and *SS*, the job order is from J_1 to J_n and the correspondence is unchangeable. Nevertheless, the job processing order in *JS* needs to be permuted. The jobs reflection sequence for *FA* and *SS* is always from I_1 to I_n which is unchangeable and the job sequence for *JS* can be changed.

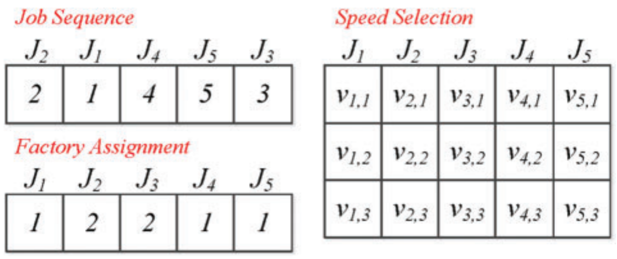
Job Sequence
J_2 J_1 J_4 J_5 J_3
2 1 4 5 3
Factory Assignment
J_1 J_2 J_3 J_4 J_5
1 2 2 1 1
Speed Selection
J_1 J_2 J_3 J_4 J_5
$v_{1,1}$ $v_{2,1}$ $v_{3,1}$ $v_{4,1}$ $v_{5,1}$
$v_{1,2}$ $v_{2,2}$ $v_{3,2}$ $v_{4,2}$ $v_{5,2}$
$v_{1,3}$ $v_{2,3}$ $v_{3,3}$ $v_{4,3}$ $v_{5,3}$

Figure 1: The solution representation of DHPFSP

Decoding schema: First, according to the *FA* vector, each job is assigned to the selected heterogeneous factories. Second, the processing sequences of all jobs in each factory are got from the *JS* vector. Then, all jobs are processed from stage one to m. Meanwhile, the actual processing time is calculated by the original processing time divided by speed $v_{i,k}$ in SS. Definitively, after all jobs are processed through all stages, the TEC and the maximum finish time (makespan) are able to be obtained.

Algorithm 1: The Framework of proposed CCSPEA.

```
Input: Total number of function evaluations ($MaxNFEs$), population
       size ($ps$), crossover rate ($P_c$), mutation rate ($P_m$)
Output: Non-dominated solutions $PF$
1  $\mathcal{P}_0 \leftarrow$ Heuristic initialization($ps$). //Initial producer population
2  $\mathcal{C} \leftarrow \varnothing$. //Initial consumer population
3  $F \leftarrow$ Decoding ($\mathcal{P}_0$). //Get fitness
4  $t = 1$.
5  while $NFEs \leqslant MaxNFEs$ do
6      //Divide population into winner and loser swarm.
7      $\mathcal{W}, \mathcal{L} \leftarrow$ Competitive Selection($\mathcal{P}_t$).
8      $\mathcal{H}_1 \leftarrow$ Cooperative evolution($\mathcal{W}, \mathcal{L}$).
9      $\mathcal{H}_2 \leftarrow$ Self-evolution($\mathcal{W}$).
10     $NFEs \leftarrow NFEs + 2 * ps$.
11     $Q_t \leftarrow \mathcal{H}_1 \cup \mathcal{H}_2 \cup \mathcal{P}_t$.
12     $\mathcal{P}_{t+1} \leftarrow$ Environmental selection ($Q_t, ps$).
13     $F_0 \leftarrow$ Get Pareto Front($\mathcal{P}_{t+1}$).
14     $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup F_0$.
15     $\mathcal{C}_{t+1} \leftarrow$ Delete duplicates ($\mathcal{C}_{t+1}$).
16     $\mathcal{C}_{t+1} \leftarrow$EnergySaving ($\mathcal{C}_{t+1}$).
17     $NFEs \leftarrow NFEs + |\mathcal{C}_{t+1}|$.
18     $\mathcal{C}_{t+1} \leftarrow$VNS ($\mathcal{C}_{t+1}$).
19     $NFEs \leftarrow NFEs + |\mathcal{C}_{t+1}|$.
20     $t = t + 1$.
21 end
22 $PF \leftarrow$ Get Pareto Front ($\mathcal{C}_t$).
```

4.3 Initialization

Initialization is a critical step for shop scheduling problems [4]. By high-quality initialization, the algorithm only pays little computation resource, and an initialization population with great convergence and diversity can be got. Algorithm 2 illustrates the detailed procedure of heuristic initialization. First, randomly generate part of *JS*, *FA*, and *SS*. Then, generate *ps*/10 *SS* with maximum speed and generate *ps*/10 *SS* with minimum speed. Finally, generate *ps*/5 *FA* by selecting a factory with the minimum workload. If several factories have the same workload, choose the factory with the minimum T. The rest of *ps*∗3/5 is supplemented by a random rule. Initializing part of solutions with min or max speed can rapidly get close to the bounds of makespan and TEC. Because the processing time of each job is the smallest when the speed selection is the max and the makespan of a solution converge to the lower bound. Similarly, the TEC converges rapidly when the speeds of all jobs are the smallest. Moreover, applying the workload balance rule can efficiently reduce makespan and makes solutions converge. Thus, those solutions initialized by heuristic rules have great convergence. Meanwhile, the rest solutions of the population are initialized by the random rule and they bring great diversity to the initialization population.

Algorithm 2: Heuristic initialization.

```
Input: population size ($ps$)
Output: producer population $\mathcal{P}(JS, FA, SS)$
Randomly generate $JS$ size $ps$.
Randomly generate $FA$ size $4/5 * ps$.
Randomly generate $SS$ size $4/5 * ps$.
for $i$= 1 to $ps/10$ do
    for $k$= 1 to $m$ do
        $v_{2i,k} \leftarrow 5.$ //max speed
    end
end
for $i$= 1 to $ps/10$ do
    for $k$= 1 to $m$ do
        $v_{1i,k} \leftarrow 1.$ //min speed
    end
end
//add up processing time of each job in each factory.
for $f$=1 to $n_f$ do
    for $i$=1 to $n$ do
        $T_{f,i} = \sum_1^m p^o_{f,i,k}.$
    end
end
for $i$= 1 to $ps/5$ do
    for $j$= 1 to $n$ do
        $FI \leftarrow$ factory index with min workload.
        if $|FI| > 1$ then
            $F_{i,j} \leftarrow$ factory index from $FI$ with min $T_{f,i}$.
        else
            $F_{i,j} \leftarrow FI.$
        end
    end
end
$SS \leftarrow SS \cup v_1 \cup v_2.$
$FA \leftarrow FA \cup F.$
```

4.4 Competitive Selection

Inspired by CSO [11] and SPEA2 [10], this work proposed a competitive selection strategy to accelerate converging. The original competitive strategy of CSO is one-to-one competition and a loser solution learns from the winner solution by differential operator shown in Eq. (1). However,

Algorithm 3: Competitive selection.

```
Input: producer population $\mathcal{P}$, Fitness $F$
Output: Winner swarm $\mathcal{W}$ and loser swarm $\mathcal{L}$
//convergence point.
for $i$= 1 to $ps - 1$ do
    for $j$= $i$+1 to $ps$ do
        if $\mathcal{P}_i \prec \mathcal{P}_j$ then
            $Dom(i, j) = 1$.
        end
        else if $\mathcal{P}_i \succ \mathcal{P}_j$ then
            $Dom(j, i) = 1$
        end
    end
end
for $i$= 1 to $ps$ do
    $R_i = \sum_{j=2}^{N} Dom(i, j)$.
end
//diversity point.
for $i$=1 to $ps$ do
    for $j$=1 to $ps$ do
        if $i \neq j$ then
            $Dis(i, j) = ||F_i - F_j||$.
        else
            $Dis(i, j) = \infty$.
        end
    end
end
for $i$= 1 to $ps$ do
    $D_i = \frac{1}{min(Dis(i,:))+2}$.
    $SF_i = R_i + D_i$. //SPEA2 Fitness.
end
$I \leftarrow$ Sort $SF$ by ascending. //competitive selection.
$\mathcal{W} \leftarrow \mathcal{P}(I(1 : ps/2), :)$.
$\mathcal{L} \leftarrow \mathcal{P}(I(ps/2 + 1 : ps), :)$.
```

DHPFSP is a discrete optimization problem and this strategy is not suitable. Thus, we designed an implicit competition according to comprehensive performance. Different from NSGA-II [44], SPEA2 has a more efficient comprehensive evaluation strategy. Thus, producer population P is divided into winner and loser swarms by fitness of SPEA2. Algorithm 3 shows the detailed procedure. First, count the domination relationship between all solutions. Then, add up the number of being dominated of each solution as its convergence point. Next, calculate the closest Euclidean distance of each solution to others and use D_i in line 20 as the diversity point of each solution. Finally, add R_i and D_i as comprehensive fitness, and sort it by ascending. The lower SF, the better performance. Thus, the first half of the sorted population is regarded as the winner swarm and the other half of the population is the loser swarm.

This strategy can rapidly assign different roles to all solutions by using its heuristic information (i.e., comprehensive fitness). After competitive selection, we design different evolutionary operators to faster converging of winner and loser swarms.

4.5 Cooperative Evolution and Self-Evolution

According to CSO [11], the loser needs to learn from the winner to accelerate convergence. In DHPFSP, the crossover operator can maximize the exchange of valid information. Thus, the quality of crossover is critical to exploration. In our approach CCSPEA, the universal crossover (UX) [45] and precedence operation crossover (POX) [1] are applied as learning operators which are shown in Fig. 2. As for POX, the job set I is randomly divided into two sets $A = \{1, 4\}$ and $B = \{2, 3, 5\}$. Then, move the jobs belonging to set A from JS_1 to the new empty solution JS_3 and move jobs from B in JS_2 to JS_4. Next, fill the empty spaces in JS_3/JS_4 by jobs of set B/A from JS_2/JS_1 in the same order. As for UX, a random 0–1 vector R size n is generated. Exchange the genes of FA_1 and FA_2 when the value of the same position $R_i = 1, i = 1, \dots, n$. Moreover, the procedures of cooperative evolution and self-evolution are shown in Fig. 3.

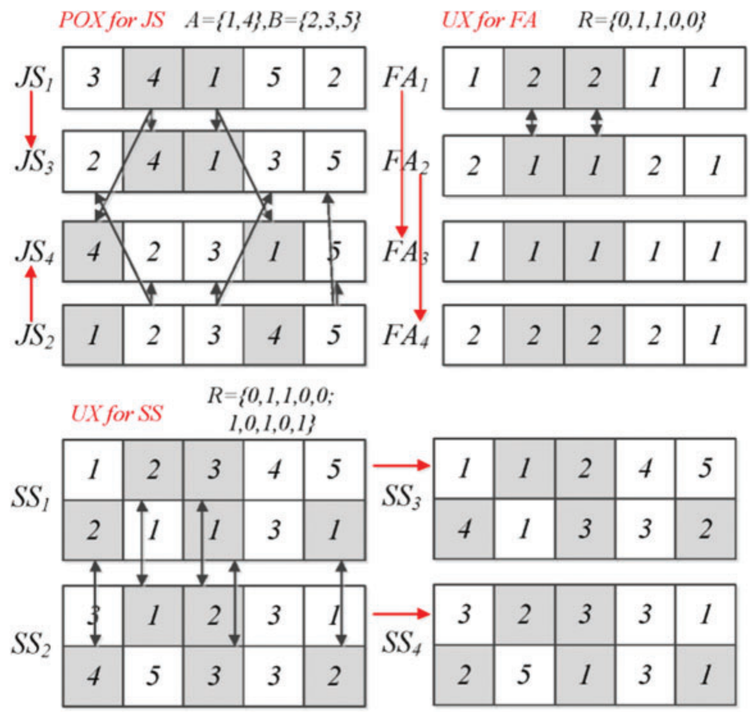

Figure 2: POX for JS, and UX for FA and SS

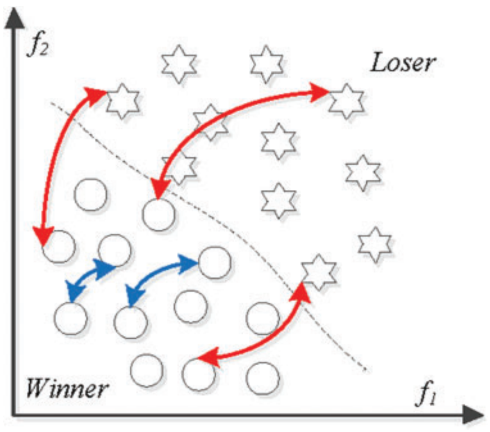

Figure 3: Cooperative evolution and self-evolution

Algorithm 4: Variable neighborhood Search (VNS).

```
Input: consumer population $\mathcal{C}$, Fitness $F$
Output: consumer population $\mathcal{C}'$
1 $l \leftarrow |\mathcal{C}|$.
2 for $i$= 1 to $l$ do
3     $CP \leftarrow$ Find the critical path of $\mathcal{C}_i$.
4     $r \leftarrow$ randomly select a number from 1 to 5.
5     if $r == 1$ then
6         $S \leftarrow$ Randomly select two job from $JS$ and exchange their
          positions.
7     end
8     if $r == 2$ then
9         $S \leftarrow$Randomly select two job from $CP$ and exchange their
          positions.
10    end
11    if $r == 3$ then
12        $S \leftarrow$Randomly select two job from $CP$ and insert the later job
          into the front of the former jobs.
13    end
14    if $r == 4$ then
15        $S \leftarrow$Randomly select a job from $CP$ and increase its speed.
16    end
17    if $r == 5$ then
18        $S \leftarrow$Randomly select a job from $CP$ and assign it to another
          factory.
```

(Continued)

Algorithm 4: Continued

```
19      end
20      if $S \prec C_i$ or not dominate each other then
21          $C \leftarrow C \cup S$
22      end
23  end
```

Cooperative evolution: To accelerate loser swarm converging, each loser will randomly select a solution from the winner swarm and execute the crossover operator.

Self-evolution: In the original CSO [11], the winner only adopts a mutation operator to update itself. However, due to the complexity of DHPFSP, each winner will execute the crossover operator by randomly selecting another winner. Moreover, each offspring will adopt three mutation operators with probability P_m to enhance diversity. As for *JS*, randomly select two jobs and exchange their positions. For *FA*, randomly select a job and move it to another factory. For *SS*, randomly select a stage of a random job and change its speed selection.

Environmental selection: To enhance convergence, the child solutions obtained from the evolution operators are combined with the population *P*. Then, select the new population from the combined population by fast non-dominated sorting and crowding distance strategy [44].

4.6 Local Search

In DHPFSP, designing neighborhood sturctures referring to problem knowledge could enhance convergence better. According to DHPFSP's features mentioned in [3]. A variable neighborhood search strategy is proposed to enhance the convergence of the consumer population. Algorithm 4 states its procedure. Firstly, search the critical path of a solution C_i. Next, randomly choose a neighborhood structure to generate offspring *S*. Finally, if the child solutions can not be dominated by the original solution, it would be added to the consumer population to improve diversity.

4.7 Energy-Saving Strategy

For green shop scheduling, the energy-saving strategy is the key procedure to lower TEC [4]. BRCE [3] has stated the problem features. This work proposed an energy-saving strategy by adjusting the speed of a solution. Algorithm 5 describes the detailed procedure of the energy-saving strategy. First, the processing flow on the first machine is determined. Second, compare $F_{i,k-1}$ and $F_{i-1,k}$ to judgment whether the idle time is generated by operation $I_{i,k-1}$ or $I_{i-1,k.}$ Then, decrease the speed of the former speed to reduce energy consumption. After traversing all operations, the TEC can be decreased.

Fig. 4 shows an example of how the energy-saving strategy reduces TEC. As for O_{22}, comparing finish time $F_{1,2}$ of O_{12} and finish time $F_{2,1}$ of O_{21} can find there is an idle gap between O_{22} and O_{12}. Thus, this is the type one that the finish time of the adjacent process on the same machine is less than the finish time of the previous operation of the same job. Moreover, reduce the speed of O_{12} and the idle time is reduced. So TEC is reduced. As for O_{32}, comparing finish time $F_{3,1}$ of O_{31} and finish time $F_{2,2}$ of O_{22} can find there is a gap that can be increased to reduce TEC and not increase the makespan. Thus, this is type two and reduce the speed of O_{31} to reduce the TEC.

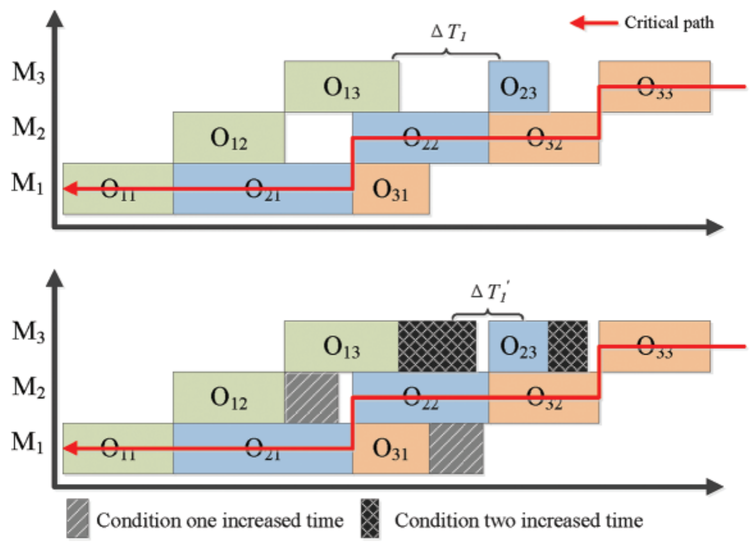

Figure 4: An example of energy-saving strategy

Algorithm 5: Energy saving strategy.

```
Input: consumer population $\mathcal{C}$, Fitness $F$
Output: consumer population $\mathcal{C}'$
1  $CP \leftarrow$ Find the critical path of $\mathcal{C}_i$.
2  for $k$= 1 to $m$ do
3      for $i$= 1 to $n$ do
4          if $k == 1$ then
5              $S_{i,k} = F_{i-1,k}$.
6          else
7              if $i == 1$ then
8                  $S_{i,k} = F_{i,k-1}$.
9              else
10                 if $F_{i,k-1} > F_{i-1,k}$ then
11                     $S_{i,k} = F_{i,k-1}$. type=1.
12                 else
13                     $S_{i,k} = F_{i-1,k}$. type=2.
14                 end
15                 if $I_{i,k}$ is in $CP$ then
16                     $F_{i,k} = S_{i,k} + p^r_{f,i,k}$.
17                     continue. //skip critical job
18                 end
19                 if $type$==1 then
```

(Continued)

Algorithm 5: Continued

```
20                  decrease $v_{i-1,k}$ until $F_{i-1,k} < S_{i,k}$ is not met. //save
                    energy
21              else
22                  decrease $v_{i,k-1}$ until $F_{i,k-1} < S_{i,k}$ is not met.
23              end
24          end
25      end
26      $F_{i,k} = S_{i,k} + p^r_{f,i,k}$.
27  end
28 end
```

5 Results of Numerical Experiment

Our approach CCSPEA is detailedly introduced in Section 4. To test the effectiveness of CCSPEA, the parameter, ablation, and comparison experiments are adopted. All algorithms are coded in MATLAB2020 and the experimental environments are on an Intel(R) Xeon(R) Gold 6246R CPU @ 3.4 GHz with 384G RAM.

5.1 Instances and Metrics

The dataset used in this work is from [3] which contains 22 different scales of instances. The number of jobs is from 20 to 200. Moreover, the amount of factories is from two to three. The machine number is from five to 20. There are totally five levels for speed selection which are $v_{f,i,k} \in \{1, 2, 3, 4, 5\}$. Moreover, the processing time of each job on the same stage is disparate in each factory. The processing power $W_O = 2.0$kWh and the idle power $W_I = 1.0$kWh. Moreover, each instance is defined by n_m_n$_f$. The stop criteria are $MaxNEFs = 400 * n \geq 2 * 10^4$.

To measure the performance of each algorithm, three types of metrics in multi-objective optimization problem are used, which are generation distance (GD) [44], Spread [44] and hypervolume (HV) [46]. The higher HV, the better the comprehensive performance. If a algorithm gets smaller GD and Spread, it has the better convergence and diversity.

5.2 Parameter Analysis Experiment

The parameter setting seriously affects an algorithm's performance for solving DHPFSP. The CCSPEA has three parameters including population size ps, mutation rate P_m of the new solution, and the starting threshold E_t of energy-saving. To simplify the parameter experiment, a design-of-experiment (DOE) Taguchi method [47] is adopted to adjust the parameter of CCSPEA. Moreover, the parameters' levels are designed following: $E_t = \{0, 0.5, 0.9\}$; $ps = \{100, 150, 200\}$; $P_m = \{0.1, 0.15, 0.2\}$. An orthogonal table L_9 (3^3) is used for parameter experiment. For a fair comparison, every parameter variant algorithm independently runs ten times under stop criteria ($MaxNFEs = 400 * n$). Moreover, the average values of HV, GD and Spread metrics for each run are recorded. Figs. 5 and 6 show three main effects plots and interaction plots of all parameters on HV, GD, and Spread metrics. Based on comprehensive observation, the best parameter configuration of CCSPEA is that $ps = 150$, $P_m = 0.15$, and $E_t = 0.9$.

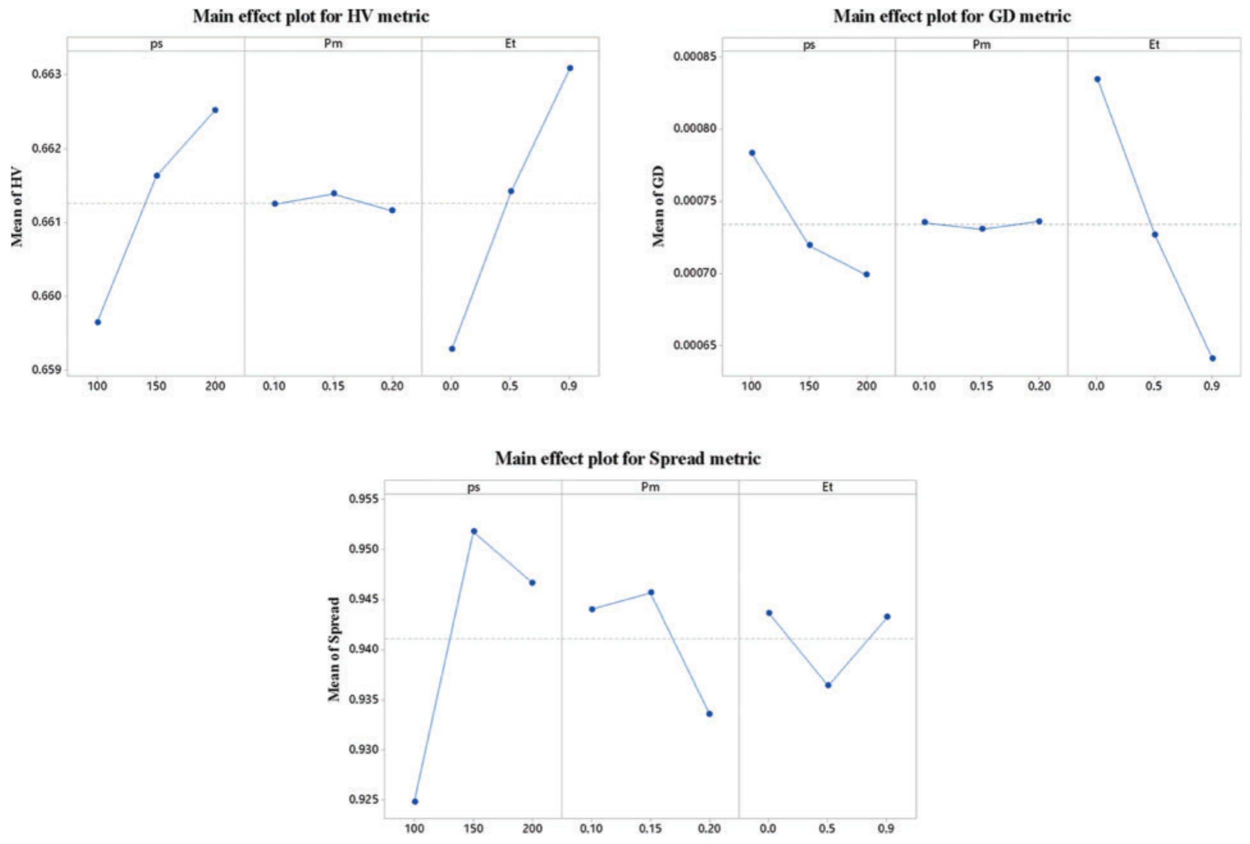

Figure 5: Main effects plots of HV, GD, and spread metrics

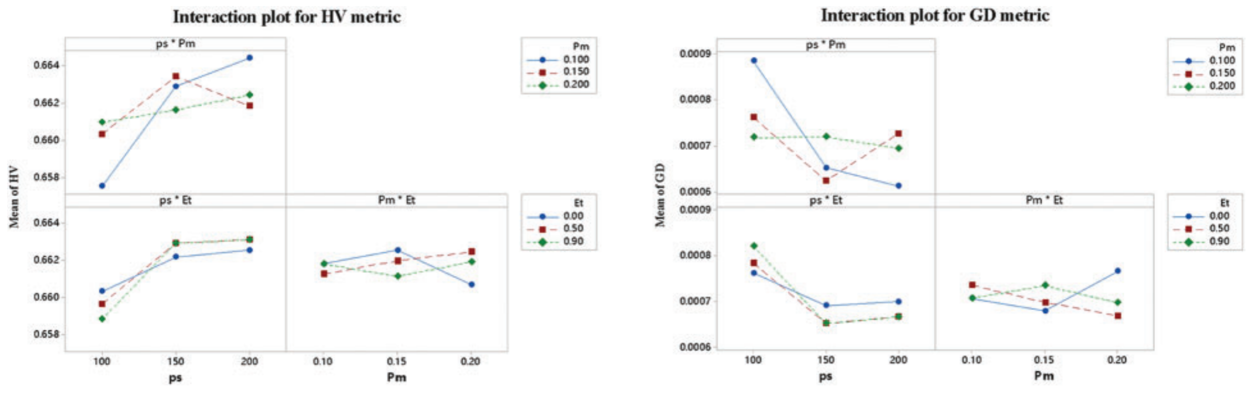

Figure 6: (Continued)

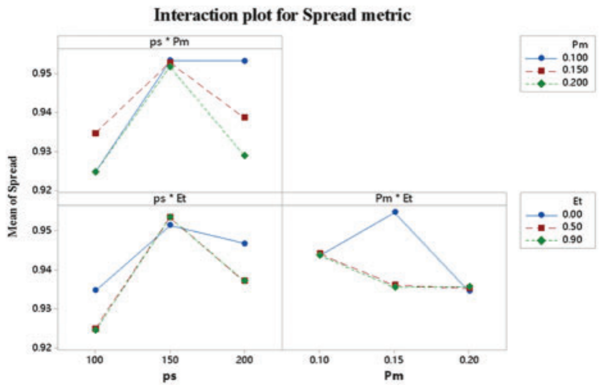

Figure 6: Interaction plots of HV, GD, and spread metrics

5.3 Components Separation Experiment

To evaluate the effectiveness of each improvement proposed in this work, four variant algorithms are created which are: i) BRCE is the original algorithm with heuristic initialization, local search, and energy-saving strategies whose effectiveness has been evaluated in [3]; ii) BRCE + P is BRCE with POX which replaces the original particle match crossover; iii) CCNSGA is BRCE + P with competitive selection and cooperative evolution. But the competitive strategy is based on fast non-dominated sorting which does not use the heuristic information of solution; vi) CCSPEA uses SPEA2 fitness to measure the comprehensive performance in CCNSGA. For a fair comparison, every variant algorithm independently executes ten times under stop criteria ($MaxNFEs = 400 * n \geq 2 * 10^4$).

Tables 2–4 state the statistical results of HV, GD, and Spread metrics of all variant algorithms. The best values of each metric are marked by **bold**. Furthermore, Table 1 shows the Friedman rank-and-sum test results with confidence level $\alpha = 0.05$. Based on the results, several conclusions are got: 1) By comparing BRCE + P and BRCE, the BRCE + P has higher rank representing that the POX crossover operator can effectively improve the convergence. 2) By comparing CCNSGA with BRCE + P, CCNSGA has higher GD and Spread rank which states that the proposed cooperative evolution can effectively improve diversity and convergence. 3) Comparing CCSPEA and CCNSGA can prove the effectiveness of the proposed competitive selection based on SPEA2 fitness. 4) The *p*-value is less than 0.05, which means a significant difference between all variants.

Table 1: The Friedman run-and-sum test results for all variant algorithms of KPMA (significant level $\alpha = 0.05$)

MOEAs	HV		GD		Spread	
	Rank	*p*-value	Rank	*p*-value	Rank	*p*-value
BRCE	3.64	1.90E-07	3.45	9.28E-07	2.55	3.15E-01
BRCE + P	2.50		2.64		2.86	
CCNSGA	2.50		2.59		2.45	
CCSPEA	**1.36**		**1.32**		**2.14**	

Table 2: Statistical results of HV (max) metric of all variant algorithms in all instances

Instances	HV							
n_m_f	BRCE		BRCE + P		CCNSGA		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.7437	0.0063	0.7447	0.0035	**0.7492**	0.0040	0.7487	0.0042
20_10_2	0.7063	0.0048	0.7100	0.0043	**0.7107**	0.0031	0.7106	0.0049
20_20_2	0.6812	0.0021	**0.6868**	0.0031	0.6857	0.0059	0.6848	0.0033
50_5_2	0.7150	0.0030	0.7191	0.0067	0.7231	0.0064	**0.7255**	0.0036
50_10_2	0.6752	0.0049	0.6789	0.0030	0.6790	0.0062	**0.6815**	0.0039
50_20_2	0.6581	0.0028	0.6596	0.0020	0.6621	0.0028	**0.6627**	0.0033
100_5_2	0.7176	0.0051	0.7177	0.0058	0.7167	0.0046	**0.7207**	0.0034
100_10_2	0.6701	0.0037	0.6720	0.0039	0.6731	0.0029	**0.6736**	0.0028
100_20_2	0.6260	0.0049	0.6258	0.0023	0.6253	0.0025	**0.6266**	0.0024
200_10_2	0.6419	0.0022	**0.6455**	0.0033	0.6421	0.0031	0.6452	0.0037
200_20_2	0.6170	0.0032	0.6161	0.0014	0.6160	0.0018	**0.6181**	0.0022
20_5_3	0.8188	0.0060	0.8202	0.0057	**0.8254**	0.0044	0.8252	0.0033
20_10_3	0.7308	0.0058	0.7324	0.0055	0.7330	0.0059	**0.7382**	0.0060
20_20_3	0.7142	0.0044	0.7162	0.0039	0.7170	0.0037	**0.7192**	0.0025
50_5_3	0.7386	0.0042	0.7430	0.0067	0.7458	0.0076	**0.7470**	0.0051
50_10_3	0.6902	0.0036	0.6942	0.0045	0.6942	0.0050	**0.6943**	0.0047
50_20_3	0.6524	0.0027	**0.6570**	0.0044	0.6564	0.0048	0.6557	0.0041
100_5_3	0.7242	0.0069	0.7280	0.0042	0.7248	0.0037	**0.7320**	0.0061
100_10_3	0.6716	0.0041	0.6747	0.0021	0.6726	0.0030	**0.6761**	0.0057
100_20_3	0.6274	0.0039	0.6248	0.0027	0.6247	0.0030	**0.6278**	0.0047
200_10_3	0.6445	0.0059	0.6465	0.0028	0.6424	0.0038	**0.6480**	0.0045
200_20_3	0.6150	0.0021	0.6166	0.0027	0.6169	0.0022	**0.6173**	0.0019

Table 3: Statistical results of GD (min) metrics of all variant algorithms in all instances

Instances	GD							
n_m_f	BRCE		BRCE + P		CCNSGA		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.0018	0.0004	0.0017	0.0003	**0.0013**	0.0003	0.0014	0.0003
20_10_2	0.0019	0.0003	0.0017	0.0003	**0.0016**	0.0002	0.0016	0.0003
20_20_2	0.0022	0.0002	**0.0017**	0.0002	0.0018	0.0005	0.0018	0.0004
50_5_2	0.0025	0.0003	0.0022	0.0006	0.0018	0.0005	**0.0016**	0.0003
50_10_2	0.0019	0.0003	0.0016	0.0002	0.0015	0.0004	**0.0014**	0.0003
50_20_2	0.0015	0.0002	0.0013	0.0001	0.0011	0.0003	**0.0011**	0.0003
100_5_2	0.0010	0.0003	0.0010	0.0004	0.0011	0.0003	**0.0008**	0.0002

(Continued)

Table 3: Continued

Instances	GD							
n_m_f	BRCE		BRCE + P		CCNSGA		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
100_10_2	0.0012	0.0003	0.0011	0.0002	0.0010	0.0002	**0.0009**	0.0001
100_20_2	0.0009	0.0002	0.0010	0.0001	0.0010	0.0001	**0.0009**	0.0001
200_10_2	0.0008	0.0001	0.0006	0.0002	0.0008	0.0001	**0.0006**	0.0001
200_20_2	0.0006	0.0001	0.0006	0.0001	0.0006	0.0001	**0.0005**	0.0001
20_5_3	0.0023	0.0004	0.0024	0.0005	0.0017	0.0005	**0.0016**	0.0005
20_10_3	0.0030	0.0005	0.0028	0.0005	0.0031	0.0009	**0.0020**	0.0006
20_20_3	0.0020	0.0004	0.0018	0.0002	0.0017	0.0003	**0.0015**	0.0003
50_5_3	0.0032	0.0006	0.0028	0.0007	0.0024	0.0007	**0.0023**	0.0006
50_10_3	0.0021	0.0004	0.0019	0.0004	0.0018	0.0004	**0.0017**	0.0002
50_20_3	0.0021	0.0003	0.0019	0.0003	**0.0018**	0.0004	0.0020	0.0003
100_5_3	0.0017	0.0004	0.0015	0.0004	0.0016	0.0003	**0.0011**	0.0004
100_10_3	0.0015	0.0003	0.0013	0.0001	0.0014	0.0002	**0.0013**	0.0003
100_20_3	0.0011	0.0003	0.0012	0.0002	0.0013	0.0002	**0.0011**	0.0003
200_10_3	0.0010	0.0003	0.0009	0.0002	0.0011	0.0002	**0.0008**	0.0002
200_20_3	0.0007	0.0001	0.0006	0.0001	0.0006	0.0001	**0.0006**	0.0001

Table 4: Statistical results of spread (min) metrics of all variant algorithms in all instances

Instances	Spread							
n_m_f	BRCE		BRCE + P		CCNSGA		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.7621	0.0682	0.7994	0.0553	**0.7391**	0.0490	0.7614	0.0358
20_10_2	0.8266	0.0480	0.8729	0.0756	0.8418	0.0489	**0.8242**	0.0517
20_20_2	0.8365	0.0705	0.8355	0.0425	0.8530	0.0565	**0.8124**	0.0390
50_5_2	0.8762	0.0355	0.8756	0.0558	0.8760	0.0591	**0.8616**	0.0371
50_10_2	0.8815	0.0495	0.8811	0.0392	**0.8736**	0.0593	0.8763	0.0481
50_20_2	0.9362	0.0831	0.9087	0.0633	0.9063	0.0329	**0.9003**	0.0752
100_5_2	0.8807	0.0459	0.8902	0.0589	0.8676	0.0446	**0.8518**	0.0397
100_10_2	0.9062	0.0661	0.9176	0.0625	0.9094	0.0388	**0.8981**	0.0502
100_20_2	0.9158	0.0541	0.9302	0.0337	**0.9142**	0.0559	0.9268	0.0448
200_10_2	**0.8921**	0.0338	0.9000	0.0281	0.9281	0.0362	0.9054	0.0331
200_20_2	0.9220	0.0586	0.9092	0.0458	0.9137	0.0419	**0.8929**	0.0318
20_5_3	**0.8605**	0.0287	0.8709	0.0549	0.8989	0.0613	0.8783	0.0290
20_10_3	0.8504	0.0677	0.8863	0.0533	**0.8222**	0.0609	0.8428	0.0536
20_20_3	**0.8526**	0.0452	0.8792	0.0542	0.8652	0.0545	0.8604	0.0501
50_5_3	0.9048	0.0443	0.8988	0.0646	0.8950	0.0989	**0.8889**	0.0986
50_10_3	0.9045	0.0556	0.9005	0.0454	0.9005	0.0482	**0.9066**	0.0721
50_20_3	0.8905	0.0298	0.9075	0.0611	0.9232	0.0542	**0.8799**	0.0713
100_5_3	**0.8784**	0.0250	0.9003	0.0646	0.9029	0.0598	0.9157	0.0505

(Continued)

Table 4: Continued

Instances	Spread							
n_m_f	BRCE		BRCE + P		CCNSGA		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
100_10_3	0.9346	0.0461	0.9365	0.0523	**0.9006**	0.0489	0.9553	0.0561
100_20_3	**0.9113**	0.0495	0.9222	0.0579	0.9457	0.0592	0.9454	0.0642
200_10_3	0.9497	0.0610	**0.9247**	0.0680	0.9352	0.0413	0.9289	0.0420
200_20_3	0.9209	0.0491	0.9231	0.0496	**0.9166**	0.0397	0.9340	0.0727

5.4 *Comparison Experiment and Discussions*

A comparison experiment is designed to further test the performance of CCSPEA, CCSPEA is compared to four state-of-arts: MOEA/D [48], PMMA [6], KCA [7], and BRCE [3]. The parameter of each comparison algorithm is set with the best parameters according to their references. As for PMMA [6], the elite rate $\beta = 0.2$, update rate $\alpha = 0.1$, and population size $ps = 40$. For MOEA/D, BRCE, and CCSPEA, the crossover rate $P_c = 1.0$, mutation rate $P_m = 0.15$, and population size $ps = 100$. For KCA [7], the population size $ps = 10$, energy-efficient rate $P_E = 0.6$, and local search times $LS = 100$. The neighborhoods updating range for MOEA/D $T = 10$. For a fair test, all MOEAs have the same stop criteria ($MaxNFEs = 400 * \mathrm{n} \geq 2 * 10^4$) and all MOEAs run 20 independent times on 22 instances.

Tables 7–9 show the statistical results of HV, GD, and Spread metrics for all MOEAs. Furthermore, the notations "$-$" and "$+$" represent that the compared algorithms are significantly worse or better than CCSPEA. Notation "=" states that there is no significant difference between compared algorithm and CCSPEA. In addition, the optimal values are marked in **bold**. Based on the results of Tables 7–9, as for HV and GD metrics, CCSPEA is significantly superior to all comparison algorithms, which means CCSPEA has better convergence and comprehensive performance than others. About the Spread metric, CCSPEA is significantly better than KCA, PMMA, and MOEA/D over twenty test problems due to its design. However, the CCSPEA has no significant difference from BRCE because the bi-roles co-evolutionary framework in BRCE can vastly improve diversity. Table 5 shows the results of Friedman rank-and-sum test of all compared MOEAs on all test problems with a confidence level $\alpha = 0.05$. CCSPEA ranks the best for HV, GD and Spread metrics and the p-value ≤ 0.05, stating that CCSPEA is significantly superior to the compared MOEAs. Table 6 shows the results of the Wilcoxon test for comparing CCSPEA to other algorithms on all metrics which are one-to-one statistical tests. The R^+/R^- means CCSPEA is better/worse than the compared algorithm. The gap between R^+ and R^- is bigger, the CCSPEA is significantly better than compared algorithm where the p-value is smaller than 0.05. Based on observation, CCSPEA is significantly superior to all MOEAs on HV metric which means CCSPEA has the best comprehensive performance. As for GD metric, CCSPEA is significantly superior to all algorithms except KCA. Because KCA has only 10 solutions and converges to the middle part of all non-dominated solutions. Thus, KCA has better convergence than CCSPEA on large-scale instances. As for Spread, CCSPEA is significantly better than all algorithms except BRCE. Because BRCE also applies the co-evolutionary framework which brings great diversity.

Table 5: The Friedman run-and-sum test results for all compared algorithms to CCSPEA (significant level $\alpha = 0.05$)

MOEAs	HV		GD		Spread	
	Rank	*p*-value	Rank	*p*-value	Rank	*p*-value
MOEA/D	3.91	9.92E-20	4.23	1.07E-18	3.27	1.90E-42
PMMA	4.73		4.00		4.18	
KCA	3.36		2.91		4.55	
BRCE	2.00		2.36		**1.50**	
CCSPEA	**1.00**		**1.50**		**1.50**	

Table 6: Results obtained by the wilcoxon test for comparing CCSPEA to other algorithms

HV				
VS	R^+	R^-	Exact *p*-value	Asymptotic *p*-value
MOEA/D	253	0	4.77E-07	0.000037
PMMA	253	0	4.77E-07	0.000037
KCA	253	0	4.77E-07	0.000037
BRCE	253	0	4.77E-07	0.000037
GD				
VS	R^+	R^-	Exact *p*-value	Asymptotic *p*-value
MOEA/D	253	0	4.77E-07	0.000037
PMMA	252	1	9.54E-07	0.000043
KCA	192	61	0.0329	0.032135
BRCE	241	12	3.34E-05	0.000189
Spread				
VS	R^+	R^-	Exact *p*-value	Asymptotic *p*-value
MOEA/D	253	0	4.77E-07	0.000037
PMMA	253	0	4.77E-07	0.000037
KCA	253	0	4.77E-07	0.000037
BRCE	138	115	$\geq$0.2	0.696841

Table 7: Statistical results of HV (max) metrics of all comparison algorithms in all instances

Instances	HV									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.5442−	0.0136	0.5106−	0.0107	0.5881−	0.0141	0.7338=	0.0065	**0.7389**	0.0043
20_10_2	0.5025−	0.0149	0.4627−	0.0155	0.5617−	0.0154	0.6912−	0.0050	**0.6956**	0.0050

(Continued)

Table 7: Continued

Instances	HV									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_20_2	0.4622−	0.0095	0.445−	0.0093	0.5224−	0.0147	0.6753=	0.0021	**0.6790**	0.0034
50_5_2	0.5459−	0.0114	0.516−	0.0125	0.5254−	0.0136	0.715−	0.0030	**0.7255**	0.0036
50_10_2	0.4784−	0.0069	0.4665−	0.0054	0.5262−	0.0096	0.6691−	0.0050	**0.6754**	0.0040
50_20_2	0.4221−	0.0048	0.4333−	0.0050	0.5181−	0.0069	0.6393−	0.0029	**0.6442**	0.0034
100_5_2	0.5451−	0.0071	0.5016−	0.0038	0.5169−	0.0073	0.7119=	0.0051	**0.7151**	0.0035
100_10_2	0.4712−	0.0071	0.4527−	0.0047	0.4988−	0.0049	0.6571−	0.0039	**0.6607**	0.0029
100_20_2	0.4106−	0.0061	0.4132−	0.0047	0.4894−	0.0050	0.6169=	0.0050	**0.6176**	0.0024
200_10_2	0.4682−	0.0039	0.4335−	0.0035	0.4738−	0.0036	0.6388−	0.0022	**0.6421**	0.0037
200_20_2	0.4161−	0.0046	0.4104−	0.0032	0.4806−	0.0052	0.617=	0.0032	**0.6181**	0.0022
20_5_3	0.5704−	0.0248	0.539−	0.0151	0.5763−	0.0293	0.7557−	0.0060	**0.7644**	0.0047
20_10_3	0.545−	0.0153	0.5086−	0.0101	0.5146−	0.0140	0.7216−	0.0058	**0.7293**	0.0061
20_20_3	0.4808−	0.0095	0.4655−	0.0116	0.541−	0.0182	0.6886−	0.0047	**0.6939**	0.0028
50_5_3	0.5658−	0.0145	0.552−	0.0127	0.5405−	0.0129	0.7386−	0.0042	**0.7470**	0.0051
50_10_3	0.5−	0.0092	0.4975−	0.0084	0.5367−	0.0111	0.6887−	0.0036	**0.6929**	0.0047
50_20_3	0.4338−	0.0104	0.448−	0.0082	0.4567−	0.0083	0.6408=	0.0027	**0.6441**	0.0042
100_5_3	0.5516−	0.0114	0.5191−	0.0090	0.4928−	0.0098	0.7043−	0.0073	**0.7125**	0.0065
100_10_3	0.4916−	0.0085	0.4808−	0.0052	0.5073−	0.0054	0.6672=	0.0041	**0.6716**	0.0058
100_20_3	0.4187−	0.0079	0.4222−	0.0032	0.498−	0.0057	0.6236=	0.0039	**0.6240**	0.0047
200_10_3	0.4845−	0.0057	0.4458−	0.0062	0.4745−	0.0054	0.6421=	0.0060	**0.6457**	0.0046
200_20_3	0.4213−	0.0067	0.4098−	0.0035	0.4873−	0.0039	0.6139=	0.0021	**0.6161**	0.0019
−/=/+	22/0/0		22/0/0		22/0/0		12/10/0			

Table 8: Statistical results of GD (min) metrics of all comparison algorithms in all instances

Instances	GD									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.0069−	0.0030	0.0134−	0.0025	0.0036−	0.0009	0.0018−	0.0004	**0.0014**	0.0003
20_10_2	0.0071−	0.0018	0.013−	0.0040	0.0025=	0.0011	0.0024=	0.0004	**0.0022**	0.0004
20_20_2	0.0061−	0.0021	0.0077−	0.0029	0.0023=	0.0006	0.0022=	0.0002	**0.0019**	0.0004
50_5_2	0.0047−	0.0019	0.0039−	0.0022	0.0062−	0.0017	0.0027−	0.0003	**0.0017**	0.0003
50_10_2	0.0048−	0.0017	0.0049−	0.0014	**0.0022=**	0.0009	0.0027−	0.0003	0.0023	0.0003
50_20_2	0.0068−	0.0022	0.0039−	0.0009	**0.0012+**	0.0005	0.0022=	0.0002	0.0020	0.0003
100_5_2	0.0025−	0.0008	0.0019=	0.0007	0.0032−	0.0007	0.0016=	0.0004	**0.0015**	0.0002
100_10_2	0.0032−	0.0010	0.0028−	0.0012	0.0022=	0.0007	0.0021=	0.0003	**0.0018**	0.0002
100_20_2	0.0038=	0.0024	0.0031−	0.0011	**0.0011+**	0.0004	0.0016=	0.0003	0.0017	0.0002
200_10_2	0.0016=	0.0010	0.0019=	0.0011	**0.0011+**	0.0004	0.0016=	0.0002	0.0015	0.0002
200_20_2	0.0025−	0.0011	0.0018=	0.0011	**0.0007+**	0.0004	0.0011=	0.0001	0.0012	0.0001
20_5_3	0.0104−	0.0020	0.0167−	0.0019	0.0076−	0.0022	0.0025−	0.0004	**0.0018**	0.0005

(Continued)

Table 8: Continued

Instances	GD									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_10_3	0.0066−	0.0033	0.0069−	0.0025	0.0066−	0.0018	0.003−	0.0005	**0.0020**	0.0005
20_20_3	0.0061−	0.0024	0.0077−	0.0023	0.0032−	0.0011	0.0022−	0.0005	**0.0016**	0.0004
50_5_3	0.0069−	0.0029	0.0053=	0.0024	0.0077−	0.0020	0.0041−	0.0005	**0.0031**	0.0005
50_10_3	0.0056−	0.0025	0.0027=	0.0011	0.0028=	0.0008	0.003−	0.0004	**0.0024**	0.0003
50_20_3	0.0058−	0.0034	0.004=	0.0031	0.0088−	0.0027	0.0031=	0.0003	**0.0029**	0.0003
100_5_3	0.0036−	0.0012	0.0034=	0.0018	0.0077−	0.0011	0.0028=	0.0004	**0.0025**	0.0005
100_10_3	0.0029=	0.0024	**0.0022**=	0.0014	0.0025=	0.0012	0.0025=	0.0002	0.0023	0.0002
100_20_3	0.0046−	0.0021	0.0038−	0.0018	**0.0011+**	0.0005	0.002=	0.0003	0.0021	0.0003
200_10_3	0.0022=	0.0012	0.0032−	0.0016	0.0023=	0.0004	0.0021=	0.0002	**0.0020**	0.0003
200_20_3	0.0021=	0.0012	0.0025−	0.0011	**0.0008+**	0.0004	0.0014=	0.0002	0.0013	0.0001
−/=/+	17/5/0		14/8/0		9/7/6		8/14/0			

Table 9: Statistical results of spread (min) metrics of all comparison algorithms in all instances

Instances	Spread									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
20_5_2	0.9549−	0.0836	0.9839−	0.0246	0.93−	0.0217	**0.7512**=	0.0741	0.7530	0.0357
20_10_2	0.9445−	0.0462	0.9976−	0.0331	0.9751−	0.0445	**0.8149**=	0.0560	0.8167	0.0567
20_20_2	0.9637−	0.0122	0.9866−	0.0128	1.0066−	0.0121	0.8373=	0.0678	**0.8139**	0.0377
50_5_2	0.9721−	0.0332	0.9927−	0.0044	0.984−	0.0176	0.8793=	0.0332	**0.8658**	0.0371
50_10_2	0.9779−	0.0199	0.9934−	0.0077	0.9907−	0.0192	0.8814=	0.0495	**0.8738**	0.0498
50_20_2	0.9892−	0.0152	0.9951−	0.0101	1.0036−	0.0151	0.9346=	0.0828	**0.8994**	0.0749
100_5_2	0.9717−	0.0276	0.9848−	0.0134	0.9864−	0.0061	0.878=	0.0479	**0.8495**	0.0415
100_10_2	0.9822−	0.0279	0.9896−	0.0101	0.9927−	0.0103	0.9071=	0.0651	**0.8994**	0.0493
100_20_2	0.9941−	0.0076	0.9925−	0.0089	1.0114−	0.0134	**0.9153**=	0.0542	0.9267	0.0443
200_10_2	0.9768−	0.0092	0.9954−	0.0055	0.996−	0.0040	**0.8914**=	0.0330	0.9049	0.0325
200_20_2	0.9923−	0.0045	0.9937−	0.0042	1.0001−	0.0037	0.9251=	0.0559	**0.8944**	0.0338
20_5_3	0.9507−	0.0665	0.9765−	0.0240	0.9233−	0.0336	**0.8046**=	0.0494	0.8249	0.0424
20_10_3	0.9533−	0.0647	0.9949−	0.0282	0.987−	0.0308	0.8496=	0.0694	**0.8419**	0.0535
20_20_3	0.9766−	0.0267	0.9936−	0.0089	1.0269−	0.0178	**0.8483**=	0.0459	0.8567	0.0517
50_5_3	0.9757−	0.0503	1.0019−	0.0283	0.9802−	0.0258	0.9046=	0.0439	**0.8896**	0.1000
50_10_3	0.9764−	0.0195	0.9913−	0.0160	0.9951−	0.0281	**0.9005**=	0.0574	0.9043	0.0721
50_20_3	0.9961−	0.0165	0.9907−	0.0077	1.0025−	0.0147	0.8897=	0.0297	**0.8783**	0.0713
100_5_3	0.9651−	0.0310	0.9929−	0.0113	0.9877−	0.0092	**0.8716+**	0.0287	0.9124	0.0542
100_10_3	0.9599=	0.0178	0.9933−	0.0063	0.9977−	0.0140	**0.9282**=	0.0523	0.9512	0.0615
100_20_3	0.9931−	0.0071	0.9918−	0.0060	1.0085−	0.0229	**0.9058**=	0.0546	0.9453	0.0687

(Continued)

Table 9: Continued

Instances	Spread									
n_m_f	MOEA/D		PMMA		KCA		BRCE		CCSPEA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
200_10_3	0.9811−	0.0135	0.994−	0.0052	0.9966−	0.0033	0.9503=	0.0595	**0.9284**	0.0436
200_20_3	0.9971=	0.0108	0.9891=	0.0080	1.0003=	0.0047	**0.9209**=	0.0491	0.9340	0.0727
−/=/+	20/2/0		21/1/0		21/1/0		0/21/1			

The success of CCSPEA is because of its design. First, the proposed POX operator has a larger step that can search more space than BRCE. Second, the designed competitive selection sufficiently uses the heuristic information (i.e., convergence, diversity, and rank) to help the solution select the crossover parent. Next, the cooperative evolution and self-evolution accelerate losers converging and let the winner further explore objective space. Finally, several efficient problem-based strategies such as heuristic initialization, VNS, and energy-saving improve the convergence. Furthermore, Fig. 7 displays the PF comparison results. Each MOEA selects the PF result with the best HV metric from 20 runs. As for the diversity and convergence of PF, CCSPEA has better Pareto solutions results than all comparison MEOAs, representing that CCSPEA has stronger ability to search Pareto solutions and get closer to practical PF.

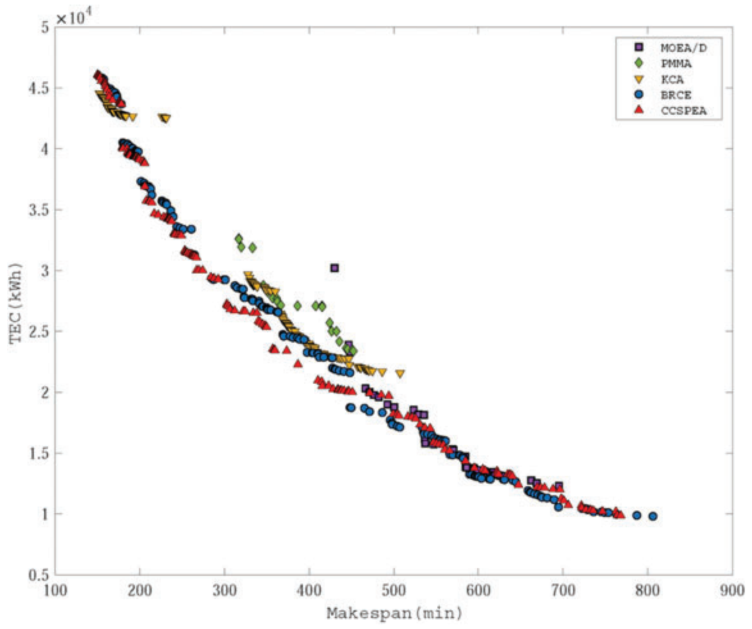

Figure 7: PF comparison results of all algorithms on 20_5_2

6 Conclusions

This work proposed a competitive and cooperative strength Pareto evolutionary algorithm for GDHPFSP. First, a competitive selection is proposed to divide the population into two swarms according to heuristic information. Second, cooperative evolution and self-evolution are designed for winner and loser swarms to accelerate convergence. Next, POX is adopted to generate a larger step to search objective space. Then, several problem-based strategies such as heuristic initialization, VNS, and energy-saving strategy are proposed to enhance exploitation. In the end, the results of detailed experiments show that CCSPEA is significantly superior to four comparison MOEAs in terms of finding PF with better diversity and convergence.

Some future tasks are discussed following: i) apply competitive strategy to other shop scheduling problems; ii) design a more efficient environmental selection strategy; and iii) consider the reinforcement learning in CCSPEA.

Funding Statement: This work was partly supported by the National Natural Science Foundation of China under Grant Nos. 62076225 and 62122093, and the Open Project of Xiangjiang Laboratory under Grant No 22XJ02003.

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

References

[1] R. Li, W. Gong, L. Wang, C. Lu and S. Jiang, "Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time," *Swarm and Evolutionary Computation*, vol. 74, pp. 101139, 2022.

[2] Y. Pan, K. Gao, Z. Li and N. Wu, "Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2022.

[3] K. Huang, R. Li, W. Gong, R. Wang and H. Wei, "BRCE: Bi-roles co-evolution for energy-efficient distributed heterogeneous permutation flow shop scheduling with flexible machine speed," *Complex & Intelligent Systems*, 2023. [Online]. Available: https://doi.org/10.1007/s40747-023-00984-x

[4] R. Li, W. Gong, C. Lu and L. Wang, "A Learning-based memetic algorithm for energy-efficient flexible job shop scheduling with type-2 fuzzy processing time," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2022.

[5] C. Lu, L. Gao, J. Yi and X. Li, "Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6687–6696, 2021.

[6] J. Chen, L. Wang, X. He and D. Huang, "A probability model-based memetic algorithm for distributed heterogeneous flow-shop scheduling," in *IEEE Congress on Evolutionary Computation (CEC)*, Wellington, New Zealand, pp. 411–418, 2019.

[7] J. -J. Wang and L. Wang, "A Knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 5, pp. 1805–1819, 2020.

[8] J. -Y. Mao, Q. -K. Pan, Z. -H. Miao, L. Gao and S. Chen, "A hash map-based memetic algorithm for the distributed permutation flowshop scheduling problem with preventive maintenance to minimize total flowtime," *Knowledge-Based Systems*, vol. 242, pp. 108413, 2022.

[9] H. Li, X. Li and L. Gao, "A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem," *Applied Soft Computing*, vol. 100, pp. 106946, 2021.

[10] E. Zitzler, M. Laumanns and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *Tech. Rep.*, 2001.

[11] X. Zhang, X. Zheng, R. Cheng, J. Qiu and Y. Jin, "A competitive mechanism based multi-objective particle swarm optimizer with fast convergence," *Information Sciences*, vol. 427, pp. 63–76, 2018.

[12] N. C. Sahoo, S. Ganguly and D. Das, “Multi-objective planning of electrical distribution systems incorporating sectionalizing switches and tie-lines using particle swarm optimization,” *Swarm and Evolutionary Computation*, vol. 3, pp. 15–32, 2012.

[13] M. Li, Z. Wang, K. Li, X. Liao, K. Hone *et al.,* “Task allocation on layered multiagent systems: When evolutionary many-objective optimization meets deep q-learning,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 842–855, 2021.

[14] X. Xu, X. Liu, Z. Xu, F. Dai, X. Zhang *et al.,* “Trust-oriented IoT service placement for smart cities in edge computing,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4084–4091, 2020.

[15] X. Xu, Q. Wu, L. Qi, W. Dou, S. B. Tsai *et al.,* “Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1787–1796, 2021.

[16] Y. Liu, X. Y. Xiao, X. P. Zhang and Y. Wang, “Multi-objective optimal STATCOM allocation for voltage sag mitigation,” *IEEE Transactions on Power Delivery*, vol. 35, no. 3, pp. 1410–1422, 2020.

[17] A. Biswas and T. Pal, “A comparison between metaheuristics for solving a capacitated fixed charge transportation problem with multiple objectives,” *Expert Systems with Applications*, vol. 170, pp. 114491, 2021.

[18] A. Biswas, L. E. Crdenas-Barrn, A. A. Shaikh, A. Duary and A. Cspedes-Mota, “A study of multi-objective restricted multi-item fixed charge transportation problem considering different types of demands,” *Applied Soft Computing*, vol. 118, pp. 108501, 2022.

[19] V. K. Maurya and S. J. Nanda, “Time-varying multi-objective smart home appliances scheduling using fuzzy adaptive dynamic spea2 algorithm,” *Engineering Applications of Artificial Intelligence*, vol. 121, pp. 105944, 2023.

[20] W. Luo, L. Shi, X. Lin, J. Zhang, M. Li *et al.,* “Finding top-k solutions for the decision-maker in multiobjective optimization,” *Information Sciences*, vol. 613, pp. 204–227, 2022.

[21] J. Cao, R. Pan, X. Xia, X. Shao and X. Wang, “An efficient scheduling approach for an iron-steel plant equipped with self-generation equipment under time-of-use electricity tariffs,” *Swarm and Evolutionary Computation*, vol. 60, pp. 100764, 2021.

[22] H. Amin-Tahmasbi and R. Tavakkoli-Moghaddam, “Solving a bi-objective flowshop scheduling problem by a multi-objective immune system and comparing with SPEA2+ and SPGA,” *Advances in Engineering Software*, vol. 42, no. 10, pp. 772–779, 2011.

[23] F. Wang, H. Zhang and A. Zhou, “A particle swarm optimization algorithm for mixed-variable optimization problems,” *Swarm and Evolutionary Computation*, vol. 60, pp. 100808, 2021.

[24] F. Wang, X. Wang and S. Sun, “A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization,” *Information Sciences*, vol. 602, pp. 298–312, 2022.

[25] Y. Tian, X. Zheng, X. Zhang and Y. Jin, “Efficient large-scale multi-objective optimization based on a competitive swarm optimizer,” *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020.

[26] X. Wang, B. Zhang, J. Wang, K. Zhang and Y. Jin, “A Cluster-based competitive particle swarm optimizer with a sparse truncation operator for multi-objective optimization,” *Swarm and Evolutionary Computation*, vol. 71, pp. 101083, 2022.

[27] Q. Gu, Y. Liu, L. Chen and N. Xiong, “An improved competitive particle swarm optimization for many-objective optimization problems,” *Expert Systems with Applications*, vol. 189, pp. 116118, 2022.

[28] W. Huang and W. Zhang, “Multi-objective optimization based on an adaptive competitive swarm optimizer,” *Information Sciences*, vol. 583, pp. 266–287, 2022.

[29] R. Cheng and Y. Jin, “A competitive swarm optimizer for large scale optimization,” *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015.

[30] P. Mohapatra, K. Nath Das and S. Roy, “A modified competitive swarm optimizer for large scale optimization problems,” *Applied Soft Computing*, vol. 59, pp. 340–362, 2017.

[31] X. Wang, K. Zhang, J. Wang and Y. Jin, “An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multiobjective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 859–871, 2022.

[32] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng *et al.,* "Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem," *Information Sciences*, vol. 619, pp. 2–18, 2023.

[33] Y. Ge, D. Chen, F. Zou, M. Fu and F. Ge, "Large-scale multiobjective optimization with adaptive competitive swarm optimizer and inverse modeling," *Information Sciences*, vol. 608, pp. 1441–1463, 2022.

[34] S. Liu, Q. Lin, Q. Li and K. C. Tan, "A comprehensive competitive swarm optimizer for large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 5829–5842, 2022.

[35] S. Qi, J. Zou, S. Yang, Y. Jin, J. Zheng *et al.,* "A Self-exploratory competitive swarm optimization algorithm for large-scale multiobjective optimization," *Information Sciences*, vol. 609, pp. 1601–1620, 2022.

[36] X. Chen and G. Tang, "Solving static and dynamic multi-area economic dispatch problems using an improved competitive swarm optimization algorithm," *Energy*, vol. 238, pp. 122035, 2022.

[37] C. He, M. Li, C. Zhang, H. Chen, X. Li *et al.,* "A competitive swarm optimizer with probabilistic criteria for many-objective optimization problems," *Complex & Intelligent Systems*, vol. 8, no. 6, pp. 4697–4725, 2022.

[38] B. H. Nguyen, B. Xue and M. Zhang, "A constrained competitive swarm optimizer with an SVM-based surrogate model for feature selection," *IEEE Transactions on Evolutionary Computation*, pp. 1, 2022.

[39] P. Musikawan, Y. Kongsorot, P. Muneesawang and C. So-In, "An enhanced obstacle-aware deployment scheme with an opposition-based competitive swarm optimizer for mobile WSNs," *Expert Systems with Applications*, vol. 189, pp. 116035, 2022.

[40] W. Shao, Z. Shao and D. Pi, "Multi-local search-based general variable neighborhood search for distributed flow shop scheduling in heterogeneous multi-factories," *Applied Soft Computing*, vol. 125, pp. 109138, 2022.

[41] F. Zhao, R. Ma and L. Wang, "A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[42] T. Meng and Q. -K. Pan, "A distributed heterogeneous permutation flow-shop scheduling problem with lot-streaming and carryover sequence-dependent setup time," *Swarm and Evolutionary Computation*, vol. 60, pp. 100804, 2021.

[43] C. Lu, L. Gao, J. Yi and X. Li, "Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6687–6696, 2021.

[44] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[45] R. Li, W. Gong and C. Lu, "A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling," *Expert Systems with Applications*, vol. 203, pp. 117380, 2022.

[46] L. While, P. Hingston, L. Barone and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

[47] R. C. Van Nostrand, "Design of experiments using the taguchi approach: 16 steps to product and process improvement," *Technometrics*, vol. 44, no. 3, pp. 289–289, 2002.

[48] Q. Zhang and L. Hui, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.