A Competitive Memetic Algorithm for Carbon-Efficient Scheduling of Distributed Flow-Shop

Jin Deng, Ling Wang(✉), Chuge Wu, Jingjing Wang, and Xiaolong Zheng

Department of Automation, Tsinghua University, Beijing 100084, China
{dengjl3, wucgl5, wang-jjl5, zhengxl11}@mails.tsinghua.edu.cn, wangling@tsinghua.edu.cn

Abstract. Considering the energy conservation and emissions reduction, carbon-efficient scheduling becomes more and more important to the manufacturing industry. This paper addresses the multi-objective distributed permutation flow-shop scheduling problem (DPFSP) with makespan and total carbon emissions criteria (MODPFSP-Makespan-Carbon). Some properties to the problem are provided, and a competitive memetic algorithm (CMA) is proposed. In the CMA, some search operators compete with each other, and a local search procedure is embedded to enhance the exploitation. Meanwhile, the factory assignment adjustment is used for each job, and the speed adjustment is used to further improve the non-dominated solutions. To investigate the effect of parameter setting, full-factorial experiments are carried out. Moreover, numerical comparisons are given to demonstrate the effectiveness of the CMA.

Keywords: Carbon-efficient scheduling · Distributed shop scheduling · Multi-objective optimization

1 Introduction

Nowadays, global warming becomes a serious public problem. Carbon dioxide (CO_2) produced during the combustion process of fossil fuel is believed to be a critical reason that causes the global warming. As fossil fuel is the main source of energy, massive amounts of CO_2 are released to the atmosphere. Realizing the danger of climate, many policies and treaties are made to restrict the emissions of greenhouse gas. About half of the world's total energy consumption is contributed by industry sector [1]. Therefore, it is imperative for manufacturing industry to implement the energy-efficient scheduling in order to reduce carbon footprints.

Under the situation, much research has been carried out about energy-saving and low-carbon. In [2], several dispatching rules were proposed relying on the estimation of inter-arrival time between jobs to control the turn on/turn off time of machines, and a multi-objective mathematical programming model was developed to minimize the energy consumption and total completion time. In [3], the turn on/off framework was applied to the single machine scheduling, and a greedy randomized multi-objective

D.-S. Huang et al. (Eds.): ICIC 2016, Part I, LNCS 9771, pp. 476–488, 2016.
DOI: 10.1007/978-3-319-42291-6_48

adaptive search algorithm was proposed to optimize total energy consumption and total tardiness. The framework was extended to the flexible flow shop scheduling [4], and genetic algorithm (GA) and simulated annealing (SA) were hybridized to minimize the makespan and the total energy consumption. Another energy saving technique is speed scaling [5]. In such a case, machines can be run at varying speeds, and the lower speed results in lower energy consumption and longer processing time. Some researchers assumed that the speed range of machine is continuous adjustable. For example, the performance of several algorithms was studied in terms of the management of energy and temperature [6]. Others considered that there are a number of discrete speeds are available for machines. Two mixed integer programming (MIP) models were presented and their performances were investigated for the permutation flow shop scheduling problem (PFSP) with makespan and peak power consumption [7]. To reduce the carbon emissions and makespan for the PFSP, an NEH-Insertion procedure was developed based on the problem properties, and a multi-objective NEH and an iterative greedy (IG) algorithm were proposed [8]. To solve the multi-mode resource constrained project scheduling with makespan and carbon emissions criteria, a Pareto-based estimation of distribution algorithm (EDA) was proposed [9].

With market dispersing throughout the world, manufacturing is changing from the traditional pattern in one single factory into the co-production among multi-factories [10]. Distributed manufacturing enables companies to improve production efficiency and profit [11]. There exist two sub-problems: allocating jobs to suitable factories and scheduling jobs on machines in each factory. Since the coupled two sub-problems cannot be solved sequentially if high performance is required, distributed scheduling is more difficult to solve [12]. Currently, the distributed PFSP (DPFSP) has been a hot topic. In [12], six MIP models and two factory assignment rules as well as several heuristics and variable neighborhood descent methods were developed. Besides, tabu search [13], EDA [14], hybrid immune algorithm [15], IG [16], and scatter search [17] have been developed to solve the DPFSP. Moreover, the DPFSP with reentrant constraint [18] have also been studied. Most of the above research only considered the time-based objectives. In this paper, the multi-objective DPFSP with carbon emissions and makespan criteria is studied. Inspired from the good performance of memetic algorithms in solving the complex optimization problems [19], we extend the competitive memetic algorithm (CMA) for the DPFSP [20] to a multi-objective version. Especially, several specific search operators are designed according to the problem characteristics. Due to the complexity brought by the optimization of the carbon emission, the sharing phase is replaced by the adjustment phase in the CMA to effectively reduce the carbon emission. In addition, some properties will be analyzed, and the effectiveness will be demonstrated by numerical comparisons.

2 Problem Description

The following notions will be used to describe the MODPFSP-Makespan-Carbon.

f: total number of factories; n: total number of jobs;

m: number of machines in each factor; s: number of speeds alternative to machines;

S: discrete set of s different processing speeds, $S = \{v_1, v_2, \ldots, v_s\}$;
n_k: number of jobs in the factory k; $O_{i,j}$: operation of job i on machine j;
$p_{i,j}$: standard processing time of $O_{i,j}$; $V_{i,j}$: processing speed of $O_{i,j}$;
$C_{i,j}$: the completion time of $O_{i,j}$; $C(k)$: the completion time of factory k;
$PP_{j,v}$: energy consumption per unit time when machine j is run at speed v;
SP_j: energy consumption per unit time when machine j is run at standby mode;
π^k: the processing sequence in the factory k; Π: a schedule, $\Pi = (\pi^1, \pi^2, \ldots, \pi^f; V)$.

The problem is described as follows. There are f identical factories, each of which is a permutation flow shop with m machines. Each of n jobs can be assigned to any one of the f factories for processing. Each operation $O_{i,j}$ has a standard processing time $p_{i,j}$. Each machine can be run at s different speeds S, and it cannot change the speed during processing an operation. When operation $O_{i,j}$ is processing at speed $V_{i,j}$, the actual processing time becomes $p_{i,j}/V_{i,j}$. Machines will not be shut down before all jobs are completed. If there is no job processed on machine j, it will be run at a standby mode. The makespan $C_{\max}$ is calculated as follows.

$$C_{\pi^k(1),1} = p_{\pi^k(1),1}/V_{\pi^k(1),1} \tag{1}$$

$$C_{\pi^k(i),1} = C_{\pi^k(i-1),1} + p_{\pi^k(i),1}/V_{\pi^k(i),1} \tag{2}$$

$$C_{\pi^k(1),j} = C_{\pi^k(1),j-1} + p_{\pi^k(1),j}/V_{\pi^k(1),j} \tag{3}$$

$$C_{\pi^k(i),j} = \max\{C_{\pi^k(i-1),j}, C_{\pi^k(i),j-1}\} + p_{\pi^k(i),j}/V_{\pi^k(i),j} \tag{4}$$

$$C(k) = C_{\pi^k(n_k),m} \tag{5}$$

$$C_{\max} = \max\{C(1), C(2), \ldots, C(f)\} \tag{6}$$

Let $x_{kjv}(t)$ and $y_{kj}(t)$ be the following binary variables:

$$x_{kjv}(t) = \begin{cases} 1, & \text{if machine } j \text{ in factory } k \text{ is run at speed } v \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$y_{kj}(t) = \begin{cases} 1, & \text{if machine } j \text{ in factory } k \text{ is run at standby mode at time } t \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The total carbon emissions (TCE) can be calculated as follows:

$$\begin{aligned} TCE &= \varepsilon \cdot TEC \\ &= \varepsilon \cdot \sum\nolimits_{k=1}^{f} \int_0^{C(k)} \left(\sum\nolimits_{v=1}^{s} \sum\nolimits_{j=1}^{m} PP_{jv} \cdot x_{kjv}(t) + \sum\nolimits_{j=1}^{m} SP_j \cdot y_{kj}(t) \right) dt \end{aligned} \tag{9}$$

where TEC is the total energy consumption and ε refers to the emissions per unit of consumed energy.

A Gantt chart of problem with 2 factories is shown in Fig. 1. Since $C(1) > C(2)$, $C_{\max} = C(1)$. Take the situation at time t_1 to explain how TCE is calculated. In the factory F_1, machine M_1 is in the standby mode, while M_2 and M_3 are in processing mode. Assuming that M_2 and M_3 are run at speed v and u, the energy consumption of factory F_1 at time t_1 is $P(t_1) = SP_1 + PP_{2v} + PP_{3u}$. Similarly, the energy consumption of each factory at different time points can be calculated as Fig. 2. The area between the curve and time axis is energy consumption of the corresponding factory. Then, TCE can be obtained by accumulating the energy consumption of each factory.

For the MODPFSP-Makespan-Carbon, it assumes that when $V_{i,j}$ is increased from v to u, the energy consumption increases while processing time decreases.

$$p_{i,j}/u < p_{i,j}/v \tag{10}$$

$$PP_{j,u} \cdot p_{i,j}/u > PP_{j,v} \cdot p_{i,j}/v \tag{11}$$

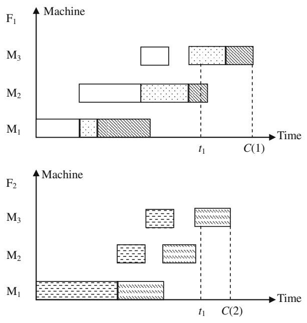

Fig. 1. Gantt chart of the DPFSP

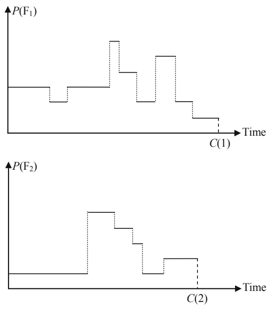

Fig. 2. Energy consumption

Based on the assumption, two properties for the PFSP were given in [8]. Here, one more property is given and extended to those of the MODPFSP-Makespan-Carbon.

Property 1 [8]. If two schedules π_1 and π_2 satisfy (1) $\forall i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,m\}$, $V_{i,j}(\pi_1) = V_{i,j}(\pi_2)$, (2) $C_{\max}(\pi_1) < C_{\max}(\pi_2)$, then, $TCE(\pi_1) < TCE(\pi_2)$. That is $\pi_1 \succ \pi_2$.

Property 2 [8]. If two schedules π_1 and π_2 satisfy (1) $C_{\max}(\pi_1) = C_{\max}(\pi_2)$, (2) $\forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,m\}$, $V_{i,j}(\pi_1) \le V_{i,j}(\pi_2)$, (3) $\exists i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,m\}$, $V_{i,j}(\pi_1) < V_{i,j}(\pi_2)$, then, $TCE(\pi_1) < TCE(\pi_2)$. That is $\pi_1 \succ \pi_2$.

Property 3. If two schedules π_1 and π_2 satisfy (1) $\forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,m\}$, $V_{i,j}(\pi_1) = V_{i,j}(\pi_2)$, (2) $TCE(\pi_1) < TCE(\pi_2)$, then, $C_{\max}(\pi_1) < C_{\max}(\pi_2)$. That is $\pi_1 \succ \pi_2$.

Proof. The TCE of π_1 and π_2 can be calculated in the following ways:

$$TCE(\pi_1) = \varepsilon \cdot (\sum_{j=1}^{m} SP_j \times t_j^{\text{idle}}(\pi_1) + \sum_{i=1}^{n} \sum_{j=1}^{m} PP_{j,V_{i,j}} \times p_{i,j}/V_{i,j}) \tag{12}$$

$$TCE(\pi_2) = \varepsilon \cdot (\sum_{j=1}^{m} SP_j \times t_j^{\text{idle}}(\pi_2) + \sum_{i=1}^{n} \sum_{j=1}^{m} PP_{j,V_{i,j}} \times p_{i,j}/V_{i,j}) \tag{13}$$

where t_j^{idle} represents the total idle time of machine M_j.

Since $TCE(\pi_1) < TCE(\pi_2)$, it has $\sum_{j=1}^{m} t_j^{\text{idle}}(\pi_1) < \sum_{j=1}^{m} t_j^{\text{idle}}(\pi_2)$. Thus, $\exists j' = 1, 2, \ldots, m$, $t_{j'}^{\text{idle}}(\pi_1) < t_{j'}^{\text{idle}}(\pi_2)$. So, $t_{j'}^{\text{idle}}(\pi_1) + \sum_{i=1}^{n} p_{i,j'}/V_{i,j} < t_{j'}^{\text{idle}}(\pi_2) + \sum_{i=1}^{n} p_{i,j'}/V_{i,j}$. According to the definition of $C_{\max}$ in a PFSP, it has $C_{\max}(\pi_1) = t_j^{\text{idle}}(\pi_1) + \sum_{i=1}^{n} p_{i,j}/V_{i,j}$, $\forall j = 1, 2, \ldots, m$. Therefore, $C_{\max}(\pi_1) < C_{\max}(\pi_2)$.

Property 4. For a schedule Π, keep the speeds of all operations unchanged and change the job processing sequence in the factory with maximum completion time (denoted as F_m). If the completion time of F_m is decreased, then the carbon emissions of F_m are also decreased. That is, if $C_{\max}(\Pi)$ is decreased, then $TCE(\Pi)$ is decreased.

Property 5. For a schedule Π, keep the speeds of all operations unchanged and change the job processing sequence in F_m. If the carbon emissions of F_m are decreased, then the completion time of F_m is also decreased. That is, if $TCE(\Pi)$ is decreased, then $C_{\max}(\Pi)$ is decreased.

Property 6. For a schedule Π, if it keeps the completion time of each factory unchanged and slows down the speeds of some operations, the $TCE(\Pi)$ will be decreased while $C_{\max}(\Pi)$ will remain the same.

3 CMA for MODPFSP-Makespan-Carbon

3.1 Encoding Scheme

In the CMA, an individual X_l is represented by a job-factory matrix ***J-F*** and a velocity matrix ***A***. ***J-F*** is a 2-by-n matrix, where the first row is job permutation sequence and the second row is factory assignment sequence. ***A*** is a n-by-m matrix, where element $A_{i,j} \in \{1, 2, \ldots, s\}$ represents the processing speed of $O_{i,j}$. An instance with $f = 2$, $n = 5$, $m = 4$, $s = 3$ is shown in Fig. 3. ***J-F*** implies that jobs J_1 and J_2 are assigned to F_1 with the processing sequence $\pi^1 = \{1, 2\}$, and jobs J_3, J_5, J_4 are assigned to F_2 with the sequence $\pi^2 = \{3, 5, 4\}$. In matrix ***A***, for example, $A_{2,3} = 3$ means that the operation $O_{2,3}$ is processed at speed v_3.

$$\mathbf{J-F} = \begin{bmatrix} 3 & 1 & 5 & 4 & 2 \\ 2 & 1 & 2 & 2 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 2 & 3 & 1 & 2 \\ 1 & 3 & 2 & 1 \\ 1 & 1 & 3 & 2 \\ 3 & 2 & 1 & 3 \\ 2 & 1 & 3 & 1 \end{bmatrix}$$

Fig. 3. An example for encoding scheme

3.2 Solution Updating Mechanism, Initialization and Archive

In the CMA, once a new individual X_l' is generated, it will be compared with its original one X_l, and the acceptance rule is based on the dominance relationship between the two individuals: (1) If $X_l' \succ X_l$, $X_l = X_l'$; (2) If $X_l \succ X_l'$, remain X_l unchanged; (3) If X_l and X_l' are non-dominated, then randomly choose one as new X_l.

In the initialization phase, all population size (*PS*) individuals are generated randomly to achieve enough diversity. Besides, a Pareto archive (PA) is used to record the explored non-dominated solutions and a temporal archive (TA) is used to store the newly found non-dominated solutions in each generation.

3.3 Competition

Adjusting the job processing sequence or the processing speed will impact the objective value, so three operators are designed, including SU, SD and CS.

SU: the operator is to increase the speed of an operation for optimizing makespan. Since the $C_{\max}$ of the DPFSP will be decreased only by improving the schedule in F_m, SU is designed to randomly choose an operation $O_{i,j}$ from the factory F_m; if $A_{i,j} = a<s$, then increase $A_{i,j}$ to b ($a < b \le s$).

SD: the operator is to decrease the speed of an operation for optimizing *TCE*. Since the reduction of carbon emissions of any factory contributes to the reduction of *TCE*, factory F_k is randomly selected in SD. Then, randomly choose an operator $O_{i,j}$ from F_k; if $A_{i,j} = a>1$, then decrease $A_{i,j}$ to b ($1 \le b<a$).

CS: based on the Properties 4 and 5, the operator is to change job processing sequence in a factory to decrease $C_{\max}$ and *TCE* simultaneously. CS is designed to randomly select a job J^* from F_m, then insert J^* into a new position in F_m.

In each generation, the objective of each individual is normalized as follows:

$$g_p(X_l) = (f_p(X_l) - f_p^{\min})/(f_p^{\max} - f_p^{\min}) \tag{14}$$

where $g_p(X_l)$ denotes the normalized value of the p-th objective, $f_p(X_l)$ denotes the value of the p-th objective, $f_p^{\max}$ and $f_p^{\min}$ denote the maximum and minimum value of the p-th objective in the current population.

It can be seen from Fig. 4 that the normalized objective space is divided into three areas by α_1, α_2 and α_3. Obviously, individuals belonging to Ω_1 has better performance on the *TCE* while is relatively weaker on the $C_{\max}$. Individuals in Ω_2 are just the opposite. Individuals in Ω_3 have better balance between the two objectives. Since individuals in Ω_1, Ω_2 and Ω_3 have different features, we make a distinction among them by choosing different operators for individuals in different areas. To be specific, individuals in Ω_1 execute operator SU to make an emphasis on optimizing $C_{\max}$, and those in Ω_2 execute operator SD to focus on the reduction of *TCE*, and those in Ω_3 carry out operator CS to optimizing both of the two objectives.

The size of the corresponding area is decided by the angles α_1, α_2 and α_3, namely, the use ranges of SU, SD and CS is controlled by the three angles. In the beginning of the CMA, we set $\alpha_1 = \alpha_2 = \alpha_3 = \pi/6$. Later, the performance of the operators may be different at different evolution phase. Besides, to adaptively adjust the use ranges of SU, SD and CS, a competition is performed among α_1, α_2 and α_3 based on the performance of operators in each generation.

To evaluate the three operators, the score of operator K denoted will be calculated after every execution. Because SU and SD are designed to optimize single objective, the score of SU or SD is calculated as follows:

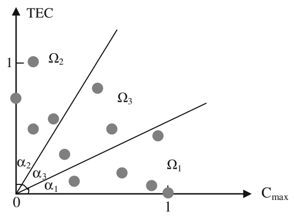

Fig. 4. Normalized objective space

```
1:  for $i = 1 \rightarrow LS$ do
2:      if $F_m = F_c$ then
3:          $CS(X^*)$
4:      else
5:          $SU(X^*)$
6:          $CS(X^*)$
7:          $SD\_2(X^*)$
8:          $CS\_2(X^*)$
9:      end if
10: end for
```

Fig. 5. Pseudocode of local search procedure

$$S_r(K) = \max\{0, f*(X_l) - f*(X_l')\}/f*(X_l) \tag{15}$$

where $f^*(X_l) = f_1(X_l)$ when K = SU, and $f^*(X_l) = f_2(X_l)$ when K = SD.

CS is to optimize both $C_{\max}$ and TCE, so its score is calculated as follows:

$$S_r(\text{CS}) = \max\{0, f_1(X_l) - f_1(X_l')\}/f_1(X_l) + \max\{0, f_2(X_l) - f_2(X_l')\}/f_2(X_l) \tag{16}$$

Let IN_q be the number of individuals in the area Ω_q. The average score of operator K is calculated as $AVS(K) = \sum_{r=1}^{IN*} S_r(K)/IN*$, where $IN^* = IN_1$ when K = SU, $IN^* = IN_2$ when K = SD, and $IN^* = IN_3$ when K = CS.

Then, the values of α_1, α_2 and α_3 are redefined as follows:

$$\alpha_q = (\pi/2 - 3\beta) * AVS(K)/\sum_K AVS(K) + \beta \tag{17}$$

where β is a small angle that guarantees $\alpha_q \neq 0$. Here, it sets $\beta = \pi/60$.

3.4 Local Intensification

It is widely recognized that local search is helpful to intensify the exploitation ability of memetic algorithms [19]. Based on the SD and CS, two more local search operators SD_2 and CS_2 are presented.

SD_2: randomly select an operation $O_{i,j}$ from the factory with maximum energy consumption (denoted as F_c), if $A_{i,j} = a > 1$, then decrease $A_{i,j}$ to b ($1 \leq b < a$).

CS_2: randomly select a job from F_c, and insert it into a new position in F_c.

In the local intensification phase, a non-dominated solution in the current population is selected to perform local search for LS times. The local search procedure which includes SU, CS, SD_2 and CS_2 is illustrated in Fig. 5. When $F_m = F_c$, only CS is performed to avoid the repeated modification of the processing speeds.

3.5 Adjustment

There are two steps in the adjustment phase. The first step is factory assignment adjustment, and the second step is speed adjustment.

In the factory adjustment, four adjusting schemes are designed. (1) Randomly select a job from factory F_m, and insert it into all possible positions of another factory. (2) Randomly select a job from factory F_m, and exchange its position with all jobs in another factory. (3) Randomly select a job from factory F_c, and insert it into all possible positions of another factory. (4) Randomly select a job from factory F_c, and exchange its position with all jobs in another factory.

Each individual chooses one of the above schemes to search better factory assignments. Let P^* be the set of non-dominated solutions that newly generated when X_l execute the factory assignment adjustment. Then, randomly select a non-dominated solution X^* from $X_l \cup P*$, and set $X_l = X^*$.

In the speed adjustment, according to the Property 5, a solution can be improved by adjusting the processing speeds without deteriorating the completion time of each factory. Therefore, the speed adjustment is performed on each solution in the TA. After the adjustment, solutions in the TA are used to update the PA. Since the completion time of a factory will not be longer if the critical path [21] remains the same, the speed adjustment is implemented on the operations that are not on the critical paths (called non-critical operations). An example of speed adjustment is shown in Fig. 6. Firstly, the critical path of each factory is found, as the arrowed line. Secondly, for each factory, the non-critical operations are selected to execute the speed adjustment from the final job back to the first. For example, the operations in Fig. 6(a) are to be adjusted in the order $\{O_{3,2} \rightarrow O_{3,1} \rightarrow O_{2,1} \rightarrow O_{1,3}\}$.

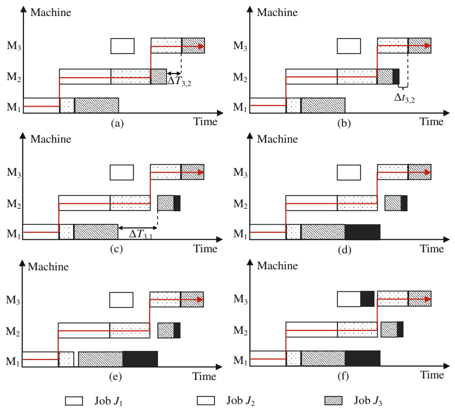

Fig. 6. Illustration of speed adjustment

Step 1: Put all the non-critical operations into the poor *Pr** in order.
Step 2: Select the first operator $O_{i,j}$ from *Pr**.
Step 3: Calculate the maximum extension time (MET) of $O_{i,j}$ as $\Delta T_{i,j} = \min\{ST_{i+1,j}, ST_{i,j+1}\} - C_{i,j}$, where $ST_{i,j}$ denotes the starting time of $O_{i,j}$, and set $ST_{n+1,j} = \infty$, $ST_{i,m+1} = \infty$. As shown in Fig. 6(a), the MET of $O_{3,2}$ is $\Delta T_{3,2}=ST_{3,3}-C_{3,2}$.
Step 4: Set $V_{i,j}$=1, 2, …, $A_{i,j}$ step by step until the condition $p_{i,j}/V_{i,j} - p_{i,j}/A_{i,j} \le \Delta T_{i,j}$ is satisfied. Then, set $A_{i,j}=V_{i,j}$, $C_{i,j} = ST_{i,j} + p_{i,j}/A_{i,j}$.
Step 5: Calculate the maximum backward time (MBT) of $O_{i,j}$ as $\Delta t_{i,j} = \min\{ST_{i+1,j}, ST_{i,j+1}\} - C_{i,j}$. As shown in Fig. 6(b), the MBT of $O_{3,2}$ is $\Delta t_{3,2}=ST_{3,3}-C_{3,2}$.
Step 6 Set $C_{i,j}=C_{i,j}+\Delta t_{3,2}$, $ST_{i,j}=ST_{i,j}+\Delta t_{3,2}$.
Step 7: Remove $O_{i,j}$ from *Pr**.
Step8: If $Pr^*\neq\emptyset$, go to **Step 2**.

Fig. 7. The procedure of speed adjustment

The procedure of speed adjustment for one factory is described as Fig. 7. And the flowchart of the CMA is illustrated in Fig. 8.

4 Computational Results

The CMA is coded in C language, and all the tests are run on the same PC with an Intel (R) core(TM) i5-3470 CPU @ 3.2 GHz/ 8 GB RAM under Microsoft Windows 7. The stopping criterion is set as $0.5 \times n$ seconds CPU time.

Since there is no benchmark for the MODOFSP-Makespan-Carbon, we generate test instances based on the test data as [8]. To be specific, f = {2, 3, 4, 5}, n = {20, 40, 60, 80, 100}, m = {4, 8, 16}, v = {1, 1.3, 1.55, 1.75, 2.10}, $p_{i,j}$ is uniformly distributed

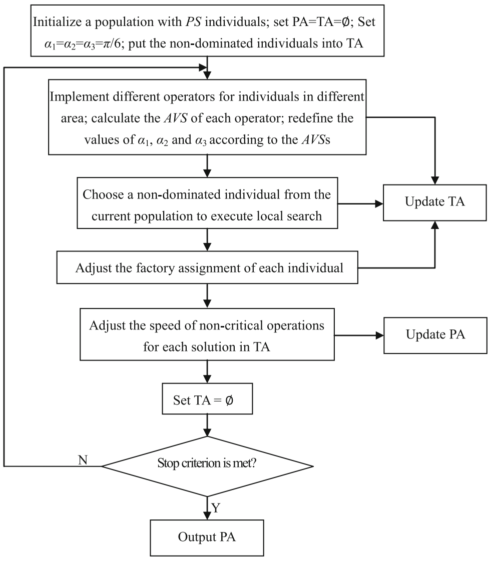
Initialize a population with PS individuals; set PA=TA=∅; Set α1=α2=α3=π/6; put the non-dominated individuals into TA
Implement different operators for individuals in different area; calculate the AVS of each operator; redefine the values of α1, α2 and α3 according to the AVSs
Choose a non-dominated individual from the current population to execute local search
Update TA
Adjust the factory assignment of each individual
Adjust the speed of non-critical operations for each solution in TA
Update PA
Set TA = ∅
Stop criterion is met?
N
Y
Output PA

Fig. 8. Flowchart of the CMA

within 5 ~ 50, $PP_{j,v} = 4 \times v^2$, $SP_j = 1$. Clearly, there are 15 combinations of $n \times m$. For each combination, 10 instances are randomly generated, and each instance is extended to $f = \{2, 3, 4, 5\}$. Thus, it has $15 \times 10 \times 4 = 600$ instances in total for evaluation.

The CMA contains two parameters: *PS* and *LS*. To investigate the influences of *PS* and *LS* on the performance of the CMA, we set *PS* with four levels {10, 20, 30, 40} and *LS* with four levels {0, 100, 200, 300}, and then 4^2 full-factorial experiments are employed. To carry out the experiments, 60 instances are generated randomly, where each corresponds to a combination of $f \times n \times m$. For each instance, 16 combinations of $PS \times LS$ are tested. For each combination of $PS \times LS$, the CMA is run 10 times independently and the obtained non-dominated solutions E_{c_i} (c_i = 1, 2, …, 16) are stored. The final non-dominated solutions *FE* are obtained by integrating E_1, E_2, …, E_{16}. Then, the contribution of a certain combination (CON) is calculated as $CON(c\_i) = \left|E'_{c\_i}\right| / |FE|$, where $E'_{c\_i} = \{X_l \in E_{c\_i} | \exists X_{l'} \in FE, X_l = X_{l'}\}$.

After all the instances are tested, the average *CON* of each combination is calculated as the response variable (RV) value. The results are listed in Tables 1 and 2, and the interval plots of *PS* and *LS* are shown in Fig. 9.

Table 1. RVs of full-factorial experiments.

Experiment Number	Factors		RV(%)	Experiment Number	Factors		RV(%)	Experiment Number	Factors		RV(%)	Experiment Number	Factors		RV(%)
	PS	*LS*			*PS*	*LS*			PS	LS			PS	LS	
1	1	1	0.0338991	5	2	1	0.0512754	9	3	1	0.0277986	13	4	1	0.0231631
2	1	2	0.0564796	6	2	2	0.0873351	10	3	2	0.0637920	14	4	2	0.0586763
3	1	3	0.0800603	7	2	3	0.1041840	11	3	3	0.0551406	15	4	3	0.0650945
4	1	4	0.0786400	8	2	4	0.0824610	12	3	4	0.0610678	16	4	4	0.0726370

Table 2. Result of analysis of variance.

Source	DF	Seq SS	Adj SS	Adj MS	F	*p*
PS	3	0.0020926	0.0020926	0.0006975	10.61	0.003
LS	3	0.0045512	0.0045512	0.0015171	23.08	0.000
Error	9	0.0005915	0.0005915	0.0000657		
Total	15	0.0072353				

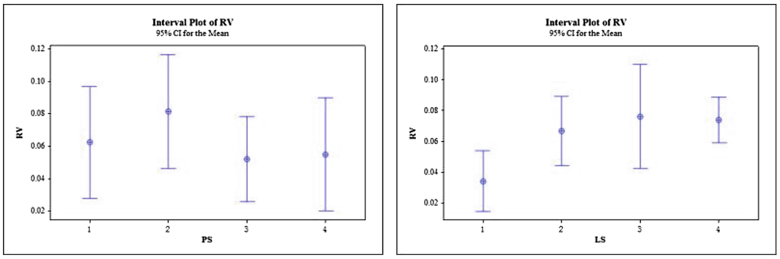

Fig. 9. Interval plot

From the Table 2, it can be seen that the influences of *PS* and *LS* are both significant with 95 % confidence interval. From Fig. 9, we know that the value of *PS* should neither be too small nor too large. A large *PS* may lead to an insufficient evolution, while a small *PS* is harmful to the diversity of the population. Similarly, a large *LS* is benefit to the exploitation, but a too large *LS* costs much of computation time on the local minima. According to the results of experiments, an appropriate combination of parameters is suggested as *PS* = 20 and *LS* = 200.

Since there is no published paper for solving the MODPFSP-Makespan-Carbon, the CMA is compared with the NSGA-II [22] and random algorithm (RA). In the NSGA-II, the population size is equal to *PS* in the CMA, and the crossover rate and mutation rate are set as 0.9 and 1/*n* as suggested in [22]. The stopping criteria of NSGA-II and RA are also set as 0.5 × *n* seconds CPU time. There are several performance metrics for multi-objective problems [23]. In this paper, we focus on the quality of the obtained non-dominated solutions. Thus, the coverage metric (CM) is used for evaluation. The CM is defined as follows:

$$C(E_1, E_2) = |\{X_2 \in E_2 | \exists X_1 \in E_1, X_2 \prec X_1 \, or X_2 = X_1\}| / |E_2| \tag{21}$$

where $C(E_1, E_2)$ denotes the percentage of the solutions in E_2 that are dominated by or the same as the solutions in E_1.

For each instance, the CMA, NSGA-II and RA are run 10 times independently within 0.5 × *n* seconds CPU time. The CM is applied to pairwise comparison between the CMA and NSGA-II as well as the CMA and RA. For the same combination of *f* × *n*×*m*, the average CM of 10 instances is calculated. The comparison results are listed in Tables 3, 4, 5 and 6 grouped by different number of *f*. From Tables 3, 4, 5 and 6, it can be seen that the proposed CMA is superior to NSGA-II and RA at all sets of instances. Besides, hypothesis testing is carried out on *C*(CMA,NSGA-II) and *C*(NSGA-II,CMA) as well as *C*(CMA,RA) and *C*(RA,CMA), and all the resulted *p*-values are equal to 0. So, it is demonstrated that the difference between *C*(CMA, NSGA-II) and *C*(NSGA-II,CMA) as well as the difference between *C*(CMA,RA) and *C*

Table 3. Comparisons of algorithms (*f* = 2).

n×*m*	CM			
	C(CMA,NSGA-II)	*C*(NSGA-II,CMA)	*C*(CMA,RA)	*C*(RA,CMA)
20×4	0.99	0.00	1.00	0.00
20×8	1.00	0.00	1.00	0.00
20×16	1.00	0.00	1.00	0.00
40×4	0.96	0.01	1.00	0.00
40×8	1.00	0.00	1.00	0.00
40×16	1.00	0.00	1.00	0.00
60×4	0.75	0.03	1.00	0.00
60×8	1.00	0.00	1.00	0.00
60×16	1.00	0.00	1.00	0.00
80×4	0.87	0.01	1.00	0.00
80×8	0.97	0.00	1.00	0.00
80×16	1.00	0.00	1.00	0.00
100×4	0.93	0.01	1.00	0.00
100×8	0.83	0.00	1.00	0.00
100×16	1.00	0.00	1.00	0.00

Table 4. Comparisons of algorithms (*f* = 3).

n×*m*	CM			
	C(CMA,NSGA-II)	*C*(NSGA-II,CMA)	*C*(CMA,RA)	*C*(RA,CMA)
20×4	1.00	0.00	1.00	0.00
20×8	1.00	0.00	1.00	0.00
20×16	0.85	0.00	1.00	0.00
40×4	1.00	0.00	1.00	0.00
40×8	1.00	0.00	1.00	0.00
40×16	1.00	0.00	1.00	0.00
60×4	1.00	0.00	1.00	0.00
60×8	1.00	0.00	1.00	0.00
60×16	1.00	0.00	1.00	0.00
80×4	0.99	0.00	1.00	0.00
80×8	1.00	0.00	1.00	0.00
80×16	1.00	0.00	1.00	0.00
100×4	0.93	0.01	1.00	0.00
100×8	1.00	0.00	1.00	0.00
100×16	1.00	0.00	1.00	0.00

Table 5. Comparisons of algorithms ($f = 4$).

$n \times m$	CM			
	C(CMA,NSGA-II)	C(NSGA-II,CMA)	C(CMA,RA)	C(RA,CMA)
20×4	1.00	0.00	1.00	0.00
20×8	1.00	0.00	1.00	0.00
20×16	0.44	0.03	1.00	0.00
40×4	1.00	0.00	1.00	0.00
40×8	1.00	0.00	1.00	0.00
40×16	0.96	0.00	1.00	0.00
60×4	1.00	0.00	1.00	0.00
60×8	1.00	0.00	1.00	0.00
60×16	1.00	0.00	1.00	0.00
80×4	1.00	0.00	1.00	0.00
80×8	1.00	0.00	1.00	0.00
80×16	1.00	0.00	1.00	0.00
100×4	0.99	0.00	1.00	0.00
100×8	1.00	0.00	1.00	0.00
100×16	1.00	0.00	1.00	0.00

Table 6. Comparisons of algorithms ($f = 5$).

$n \times m$	CM			
	C(CMA,NSGA-II)	C(NSGA-II,CMA)	C(CMA,RA)	C(RA,CMA)
20×4	1.00	0.00	1.00	0.00
20×8	0.95	0.00	1.00	0.00
20×16	0.25	0.16	0.97	0.00
40×4	1.00	0.00	1.00	0.00
40×8	1.00	0.00	1.00	0.00
40×16	0.97	0.00	1.00	0.00
60×4	1.00	0.00	1.00	0.00
60×8	1.00	0.00	1.00	0.00
60×16	1.00	0.00	1.00	0.00
80×4	1.00	0.00	1.00	0.00
80×8	1.00	0.00	1.00	0.00
80×16	1.00	0.00	1.00	0.00
100×4	1.00	0.00	1.00	0.00
100×8	1.00	0.00	1.00	0.00
100×16	1.00	0.00	1.00	0.00

(RA,CMA) are significant with 95 % confidence interval. Thus, it is concluded that the CMA is more effective than the NSGA-II and RA in terms of the quality of the obtained solutions.

5 Conclusions

This is the first work to consider the carbon-efficient scheduling for the distributed permutation flow shop scheduling problem with makespan and total carbon emissions criteria. Some properties were analyzed, a competitive memetic algorithm was proposed, the effect of parameter setting was investigated, and the effectiveness of the designed CMA was demonstrated. Future work could focus on the design of the new search operators and new mechanisms to perform competition. It is also interesting to studying the carbon-efficient scheduling for other distributed scheduling problems.

Acknowledgement. This research is supported by the National Key Basic Research and Development Program of China (No. 2013CB329503) and the National Science Fund for Distinguished Young Scholars of China (No. 61525304).

References

1. Fang, K., Uhan, N., Zhao, F., Sutherland, J.W.: A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. J. Manuf. Syst. **30**, 234–240 (2011)
2. Mouzon, G., Yildirim, M.B., Twomey, J.: Operational methods for minimization of energy consumption of manufacturing equipment. Int. J. Prod. Res. **45**, 4247–4271 (2007)
3. Mouzon, G., Yildirim, M.B.: A framework to minimise total energy consumption and total tardiness on a single machine. Int. J. Sustain. Eng. **1**, 105–116 (2008)
4. Dai, M., Tang, D., Giret, A., Salido, M.A., Li, W.D.: Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robotics Comput.-Integr. Manuf. **29**, 418–429 (2013)

5. Yao F., Demers A., Shenker S.: A scheduling model for reduced CPU energy. In: 36th Annual Symposium on Foundations of Computer Science, pp. 374–382 (1995)
6. Bansal, N., Kimbrel, T., Pruhs, K.: Speed scaling to manage energy and temperature. J. ACM **54**, 3 (2007)
7. Fang, K., Uhan, N.A., Zhao, F., Sutherland, J.W.: Flow shop scheduling with peak power consumption constraints. Ann. Oper. Res. **206**, 115–145 (2013)
8. Ding, J.Y., Song, S., Wu, C.: Carbon-efficient scheduling of flow shops by multi-objective optimization. Eur. J. Oper. Res. **248**, 758–771 (2016)
9. Zheng, H., Wang, L.: Reduction of carbon emissions and project makespan by a Pareto-based estimation of distribution algorithm. Int. J. Prod. Econ. **164**, 421–432 (2015)
10. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. Springer, Berlin (2012)
11. Wang, B.: Integrated Product, Process and Enterprise Design. Chapman & Hall, London (1997)
12. Naderi, B., Ruiz, R.: The distributed permutation flowshop scheduling problem. Comput. Oper. Res. **37**, 754–768 (2010)
13. Gao, J., Chen, R., Deng, W.: An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. Int. J. Prod. Res. **51**, 641–651 (2013)
14. Wang, S., Wang, L., Liu, M., Xu, Y.: An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. Int. J. Prod. Econ. **145**, 387–396 (2013)
15. Xu, Y., Wang, L., Wang, S., Liu, M.: An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem. Eng. Optim. **46**, 1269–1283 (2014)
16. Fernandez-Viagas, V., Framinan, J.: A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. Int. J. Prod. Res. **53**, 1111–1123 (2015)
17. Naderi, B., Ruiz, R.: A scatter search algorithm for the distributed permutation flowshop scheduling problem. Eur. J. Oper. Res. **239**, 323–334 (2014)
18. Rifai, A.P., Nguyen, H.T., Dawal, S.Z.M.: Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl. Soft Comput. **40**, 42–57 (2016)
19. Ong, Y.S., Lim, M., Chen, X.: Research frontier-memetic computation-past, present and future. IEEE Comput. Intell. Mag. **5**, 24–31 (2010)
20. Deng J., Wang L., Wang S.: A competitive memetic algorithm for the distributed flow shop scheduling problem. In: 2014 IEEE International Conference on Automation Science and Engineering, pp. 107–112. IEEE Press, New York (2014)
21. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flow-shop problem. Eur. J. Oper. Res. **91**, 160–175 (1996)
22. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comput. **6**, 182–197 (2002)
23. Li, B., Wang, L., Liu, B.: An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**, 818–831 (2008)