Future Generation Computer Systems 113 (2020) 441–453

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A multi-objective evolutionary algorithm based on adaptive clustering for energy-aware batch scheduling problem

Si-yuan Qian [a,b], Zhao-hong Jia [a,b,c,*], Kai Li [d]

[a] Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University, Hefei, Anhui, 230601, PR China
[b] School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, PR China
[c] School of Internet, Anhui University, Hefei, Anhui, 230601, PR China
[d] School of Management, Hefei University of Technology, Hefei, Anhui, 230009, PR China

ARTICLE INFO

Article history:
Received 7 September 2019
Received in revised form 28 April 2020
Accepted 6 June 2020
Available online 3 July 2020

Keywords:
Multi-objective optimization
Evolutionary algorithm
Adaptive clustering
Energy-aware scheduling
Batch processing machines

ABSTRACT

For batch scheduling problems, more and more attentions have been paid to reducing energy consumption. In this paper, a complex batch scheduling problem on parallel batch processing machines considering time-of-use electricity price is investigated to minimize makespan and total electricity cost, simultaneously. Due to NP-hardness of the studied problem, a multi-objective evolutionary algorithm based on adaptive clustering is proposed, where an improved adaptive clustering method is incorporated to mine the distribution structure of solutions, which can be used to guide the search. Moreover, a new recombination strategy based on both distribution characteristics and mating probability is designed to select individuals for mating. In addition, to better balance exploration and exploitation, the mating probability is adaptively adjusted according to historical information. The experimental results demonstrate the competitiveness of the proposed algorithm in terms of solution quality.

1. Introduction

Scheduling models widely exist in reality since any problem related to allocating resources is inseparable from scheduling. Nowadays, with the rapid development of computer industries, more and more scheduling problems have appeared in the fields related to computer science and technology, such as workflow scheduling under cloud environment [1,2], task scheduling in edge computing [3], concurrent container scheduling in distributed computation [4], and so on. In addition, scheduling is closely related to real production. Classical scheduling and batch scheduling are two typical categories of production scheduling. In classical scheduling, each machine is assumed to process only one job at any time, and each job can be processed on at most one machine at any time. As one type of modern scheduling, batch scheduling problem (BSP) originates from the final production stage of semiconductor chips. In BSP, multiple jobs can be processed on one batch machine, simultaneously, which is the major difference from classical scheduling. Batch scheduling generally includes two sub-problems. That is, how to choose jobs to generate batches under the constraint of machine capacity, and how to schedule the batches on BPMs. After the two sub-problems are addressed, a schedule can be generated so as to obtain the objective value. At present, the realistic cases of BSPs include compact strip production [5], heat treatment [6], glass manufacturing [7], paper mill industry [8], and etc.

Recently, a number of researchers have paid attentions to BSPs with optimizing energy consumption, since environmental protection and energy consumption have become important issues of manufacturing. To achieve sustainable development, it is preferable for manufacturing factories to solve the BSP with energy consumption to not only boost the economic benefits but also conserve energy. The electricity is widely applied in manufacturing so that minimizing power consumption, which can be measured by total electricity cost, has become one of the hotspots of manufacturing industries. Moreover, electricity prices generally vary with time in most areas, which makes it possible to shift the electricity load to decrease energy consumption during the production. Furthermore, it is of great significance to investigate the BSPs with energy consumption because the total electricity cost is able to be reduced without deteriorating the completion time of batches.

In this paper, we investigate a multi-objective BSP of minimizing makespan and total electricity cost simultaneously. Since most BSPs are NP-hard, meta-heuristic algorithms have been effectively employed in solving BSPs. In addition, multi-objective evolutionary algorithms (MOEAs) have shown good performance

* Corresponding author at: School of Computer Science and Technology, Anhui University, Hefei, Anhui, 230601, PR China.
E-mail address: zhjia@mail.ustc.edu.cn (Z.-h. Jia).

https://doi.org/10.1016/j.future.2020.06.010
0167-739X/© 2020 Published by Elsevier B.V.

in addressing multi-objective optimization problems (MOPs). For MOPs, the distribution characteristics hidden in the solutions often imply the features of the solutions to the problem, as well as the characteristics of search space. Therefore, it is favorable for MOEAs to attract and utilize the distribution characteristics of solutions in the evolutionary process. Although clustering methods can be used to attract the distribution characteristics, it is generally required that the number of clusters is determined in advance, whose disadvantage is that clustering results depend on the priori knowledge completely. As illustrated in Fig. 1(a), the population is clustered into two clusters when the number of clusters is set to two in advance. When the population evolves into the status shown as Fig. 1(b), two new clusters are obtained by the classical clustering method with the predetermined number of clusters. However, as shown in Fig. 1(b) and (c), it is more accurate when the population is grouped into three clusters. Therefore, in order to obtain the distribution characteristics that coincides with the real evolutionary trend as much as possible, the number of clusters should be adaptively adjusted based on the objective space of the current population. Base the above considerations, a MOEA based on adaptive clustering is proposed to tackle this problem. The main contributions of this study are summarized as follows:

(1) An improved adaptive clustering method is presented to extract the distribution characteristics of solutions in search space. The mined characteristics are then used to direct the evolution. Moreover, in the adaptive clustering, the number of clusters is redetermined every certain number of iterations and the iterative process of clustering is integrated into the evolutionary process to reduce time consumption.

(2) In most studies, the individuals are selected randomly for mating so that the changeable needs for exploration and exploitation during the search is hard to be satisfied. Hence, a new recombination restriction strategy is developed to select appropriate mating individuals, according to the distribution characteristics and mating probability simultaneously. As shown in Fig. 1(d), recombination mating of individuals from the same cluster can enhance the exploitation, while recombination mating of those from different clusters can improve the exploration.

(3) Considering that mating needs of individuals are variable during evolution, the mating probability is designed to be adaptively adjusted. The greater (smaller) the mating probability, the more it is inclined to exploitation (exploration). That is, based on historical information, an adaptive adjustment of mating probability is presented to balance exploitation and exploration effectively.

The rest of this paper is organized as follows: The literature review is discussed in Section 2. Section 3 defines the studied problem. The proposed algorithm is described in Section 4. Section 5 depicts the experimental designs and analyzes the results of comparative experiments. Section 6 summarizes this paper and points out some future research directions.

2. Literature review

In this section, we first review related work to BSPs considering energy consumption. Then, the MOEAs combined with clustering are introduced.

2.1. Energy-aware batch scheduling

In the BSP, first proposed by Ikura and Gimple [9], the jobs have the same size and the size of each batch is determined by the number of jobs assigned in the batch. Then, Uzsoy [10] extended the problem to a single BPM with non-identical jobs, to minimize makespan and total completion time, respectively. Lee and Uzsoy [11] investigated the BSP on a single BPM with dynamic job arrivals to minimize makespan. For an extensive review for the BSPs, the readers are referred to the articles [12] and [13]. In order to be closer to real world applications, researchers have paid more attentions on the BSPs with more complicated constraints and diversified objectives.

In the past, the optimized objectives of BSPs were related to production efficiency, such as: makespan, total completion time, total delay time, etc. With the arising of green manufacturing, there has been a growing number of studies on BPMs scheduling to save electricity cost under time sensitive electricity prices. Cheng et al. [14] considered the problem of minimizing makespan and total electricity cost simultaneously on one BPM with the time-of-use (TOU) electricity price, and proposed an ϵ-constrained method to find the Pareto front of the solutions. With the consideration of switching machines under TOU tariffs, a more complicated BSP was investigated in [15], and a heuristic algorithm based on ϵ-constraint was designed to solve large-scale problems. In addition, Zhang et al. [16] also addressed bi-objective single-machine batch scheduling under time-of-use electricity prices to minimize total energy cost and makespan simultaneously. With the development of intelligent algorithms, Jia et al. [17] proposed a multi-objective ant colony optimization (ACO) algorithm to resolve a bi-criteria BSP of minimizing makespan and total electricity cost on parallel BPMs with the same processing speed, where two local optimization strategies were incorporated to improve solution quality. Extending to study the problem on parallel BPMs with different speeds, Zhou et al. [18] presented a differential evolution (DE) algorithm with a new DE operator to minimize makespan and total electricity cost at the same time, without considering non-identical job sizes. Tan et al. [19] proposed genetic algorithms with greedy strategy to minimize total electricity costs on non-identical parallel machines. Zheng et al. [20] designed a hybrid ACO algorithm to optimize makespan and total electricity cost in a two-stage flow shop. Additionally, Zeng et al. [8] proposed a hybrid NSGA-II algorithm to minimize three objectives, i.e., makespan, total electricity cost, and material consumption, in a flexible flow shop with BPMs, which has been applied to solve real scheduling problem. Liu et al. [21] investigated a bi-objective hybrid flow shop problem of the tempered glass scheduling, and proposed a DE algorithm to optimize makespan and total electricity cost, simultaneously. Furthermore, problems considering the other evaluation criteria related to energy consumption, such as minimizing carbon dioxide emissions [22] and minimizing heat loss in heat treatment [6], have also been investigated.

In summary, studies on BSPs considering electricity cost, especially on BSPs with different processing speeds and non-identical job sizes, are still much less than those of traditional objectives. Since energy consumption existing in real production widely, the problems considering electricity cost are still worth further study.

2.2. Clustering-based MOEAs

MOEAs are search algorithms that simulate natural selection and evolution. According to the criteria of fitness evaluation, MOEAs can be roughly categorized into three classifications: (1) The first is known as the Pareto-based approaches, such as NSGA-II [23], SPEA2 [24], and so on; (2) The second, based on decomposition, decomposes a MOP into a group of subproblems, such as MOEA/D [25], IMOEA/D [26], and so on; (3) The third adopts performance indicators as the selection criterion, such as IBEA [27], HypE [28], and so on.

Additionally, there are MOEAs combined with other optimization mechanisms, called variants of MOEAs, one of which is

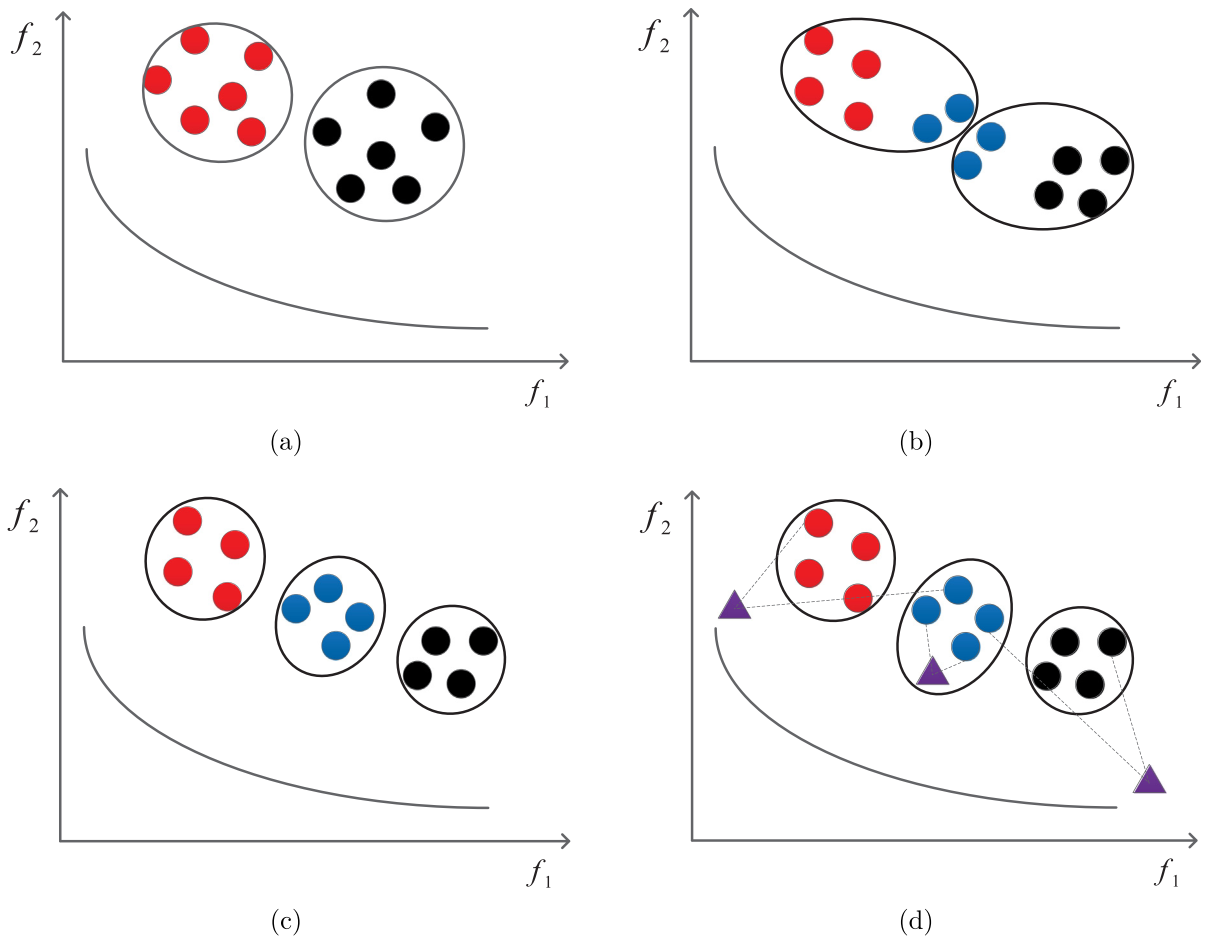

Fig. 1. (a) Initial clustering results. (b) Results of classical clustering method. (c) Results of the proposed clustering method. (d) Comparison of two proposed recombination mating methods.

combined with clustering strategy. There are mainly five aspects where clustering can be incorporated into MOEAs. First, the clustering is used in environmental selection. Gong et al. [29] proposed an adaptive partitioning clustering to select solutions for the next generation. Spea et al. [30] adopted clustering to manage external archive in differential EA. Hua et al. [31] adaptively chose solutions to maintain diversity of population in the key layer during the process of non-dominated sorting. Wang et al. [32] adopted fuzzy clustering to select the Pareto-optimal solutions for the final decision. Second, the clustering is applied to recombination restriction. Based on the clustering, Kotinis et al. [33] selected adjacent individuals to cross in each cluster. Zhang et al. [34] designed a recombination mating strategy based on the adaptive clustering to address continuous optimization problems. Third, new solutions are generated through the probability model which is based on clustering. Zhang et al. [35] built a Gaussian process model based on the fuzzy clustering. Sun et al. [36] constructed a new Gaussian sampling model based on the K-means clustering. Fourth, the population is divided into different sub-populations which are evolved independently by using clustering. Zhang et al. [37] utilized clustering to divide the large-scale decision variables into two types, i.e., convergence-related variables and diversity-related variables, which are optimized separately. Wang et al. [38] employed clustering to partition the population and each sub-population completed reproduction independently. Fifth, clustering is applied to other aspects. Lee et al. [39] adopted fuzzy clustering to reduce the number of evaluation in solving graph coloring problem. Cai et al. [40] employed clustering algorithm in generating reference points for many-objective optimization problems (MaOPs).

Based on the above discussion, it can be found that clustering is able to be combined with MOEAs in different ways. Moreover, most studies focus on environmental selection. However, the true Pareto fronts of most multi-objective BSPs are unknown. Therefore, it is important to generate solutions that are close to the true Pareto front during reproduction. Furthermore, it can be seen that better new solutions can be constructed to multi-objective BSPs through using recombination restriction strategy.

3. Problem definition

According to the three field notation [41], the studied multi-objective BSP can be denoted as $Q_m \mid r_j, p_j, s_j, \mathrm{C} \mid (C_{\max}, TEC)$. After the problem is described, the mathematical model is provided.

3.1. Problem description

The assumptions of the studied problem are described as follows.

(1) There are n jobs to be processed on m parallel BPMs. Each job j ($j = 1, 2, \ldots, n$) has three attributes: release time r_j, processing time p_j, and size s_j. Each machine i ($i = 1, 2, \ldots, m$) has four attributes: machine capacity C, processing speed v_i, processing power PP_i, and standby power SP_i.

(2) The jobs are grouped into batches that are processed on machines. The generated batch set is denoted by B. The bth

batch processed on machine i is denoted by B_{bi} whose processing time, denoted by PT_{bi}, is determined by the longest actual processing time among those of all the jobs in B_{bi}. The actual processing time of job j on machine i equals to $\lceil p_j/v_i \rceil$. The release time RT_{bi} is given by the largest release time of all the jobs in B_{bi}, i.e., $RT_{bi} = \max\{r_j | j \in B_{bi}\}$. The size of a batch equals to the total sizes of all the jobs in this batch. There is a constraint of machine capacity. That is, the size of each batch cannot exceed the capacity of the machine that processes this batch, i.e., $\sum_{j \in B_{bi}} s_j \le C$.

(3) The start processing time of batch B_{bi} denoted by ST_{bi} depends on the larger of the release time of B_{bi} and the completion time of the previous batch on the same machine, denoted by $CT_{b-1,i}$, i.e., $ST_{bi} = \max\{RT_{bi}, CT_{b-1,i}\}$. The completion time of B_{bi}, denoted by CT_{bi}, is defined as the sum of the start processing time and the processing time of B_{bi}, i.e., $CT_{bi} = ST_{bi} + PT_{bi}$.

(4) Each job can only be assigned to one batch on one machine. The batch is not allowed to be interrupted during its processing. The completion time of the 0th batch on each machine is set to zero.

(5) The first optimization objective is to minimize makespan, denoted by $C_{\max}$, and the second optimization objective is to minimize total electricity cost, denoted by TEC, which is calculated based on the time-of-use electricity price.

3.2. Mathematical model

Based on the above assumptions, the mathematical model of the studied problem is formulated as follows.

$$\text{Minimize} \quad C_{\max} \tag{1}$$

$$\text{Minimize} \quad TEC = \sum_{i=1}^{m} \left(SP_i \sum_{b=0}^{k-1} \sum_{t=CT_{bi}+1}^{ST_{b+1,i}} f(t) + PP_i \sum_{b=1}^{k} \sum_{t=ST_{bi}+1}^{CT_{bi}} f(t) \right) \tag{2}$$

$$\text{Subject to:} \quad \sum_{i=1}^{m} \sum_{b=1}^{k} X_{jbi} = 1 \quad j = 1, \dots, n \tag{3}$$

$$\sum_{j=1}^{n} s_j X_{jbi} \le C \quad b = 1, \dots, k; i = 1, \dots, m \tag{4}$$

$$RT_{bi} \ge r_j X_{jbi} \quad j = 1, \dots, n; b = 1, \dots, k; i = 1, \dots, m \tag{5}$$

$$PT_{bi} \ge \lceil p_j X_{jbi}/v_i \rceil \quad j = 1, \dots, n; b = 1, \dots, k; i = 1, \dots, m \tag{6}$$

$$CT_{bi} = ST_{bi} + PT_{bi} \quad b = 1, \dots, k; i = 1, \dots, m \tag{7}$$

$$ST_{bi} \ge CT_{(b-1)i} \quad b = 1, \dots, k; i = 1, \dots, m \tag{8}$$

$$ST_{bi} \ge RT_{bi} \quad b = 1, \dots, k; i = 1, \dots, m \tag{9}$$

$$CT_{0i} = 0 \quad i = 1, \dots, m \tag{10}$$

$$C_{\max} \ge CT_{bi} \quad b = 1, \dots, k; i = 1, \dots, m \tag{11}$$

$$X_{jbi} \in \{0, 1\} \quad j = 1, \dots, n; b = 1, \dots, k; i = 1, \dots, m \tag{12}$$

Objective function (1) is to minimize $C_{\max}$. Objective function (2) is to minimize TEC, where $f(t)$ denotes electricity price during the tth unit time. Constraint (3) ensures that each job can only be assigned to one batch on one machine. Constraint (4) ensures the total sizes of all the jobs in one batch cannot exceed the capacity of the machine that processes this batch. Constraint (5) indicates that the release time of the batch depends on the largest release time among all the jobs in this batch. Constraint (6) specifies that the processing time of the batch is given by the longest actual processing time among all the jobs in this batch. Constraint (7) defines the completion time of the batch. Constraint (8) guarantees that the process of batch is not allowed to be interrupted. Constraint (9) ensures that the batch cannot be processed until its release time. Constraint (10) indicates that the completion time of the 0th batch on each machine is zero. Constraint (11) defines the makespan. Constraint (12) indicates that the decision variable X_{jbi} is a binary variable. $X_{jbi} = 1$ denotes job j is assigned to the bth batch on machine i; Otherwise, job j is not assigned to the bth batch processed on machine i.

4. Proposed algorithm

The proposed Pareto-based MOEA is called the adaptive clustering-based evolutionary algorithm (ACBEA). The most obvious difference between ACBEA and the other original Pareto-based MOEAs is that an improved adaptive clustering method is incorporated into ACBEA, which can extract the distribution characteristics of solutions in search space. The details of the proposed algorithm are elaborated in the following sections.

4.1. Framework of the proposed algorithm

The framework of ACBEA is described as Algorithm 1. In lines 1–2, after both the current iteration t and an empty non-dominated set A are initialized, an initial population O_t is generated. In line 4, the clustering result Cr is initialized. In line 5, the current population O_t is partitioned into K clusters by using the adaptive clustering method, and the clustering result Cr is obtained. In line 6, a reproduction operator is applied to generate a set of N new individuals, denoted by Q. In line 7, an environmental selection operator based on fast non-dominated sorting proposed in [23] is employed to select N individuals from O_t and Q to generate the next population O_{t+1}. In line 8, the mating probability is adjusted according to the historical information. In line 9, non-dominated solutions in O_t and Q are chosen to update non-dominated set A according to Pareto-dominance relationship. This procedure repeats until the termination condition is satisfied.

Algorithm 1 ACBEA

```
Input: Population size $N$; maximum iteration $T$; initial mating
    probability $P_1$; historical iteration $L$; updated iteration $H$.
Output: Non-dominated set $A$.
1: $t \leftarrow 1, A \leftarrow \varnothing$;
2: Initialize population $O_t$;
3: while $t \le T$ do
4:     $Cr \leftarrow \varnothing$; /*Initialize clustering result.*/
5:     $Cr \leftarrow$ AC($O_t$,$t$,$H$); /*$O_t$ is partitioned into $K$ clusters.*/
6:     $Q \leftarrow$ REP($O_t$,$P_t$,$Cr$); /*Generate the set of new individuals
       $Q$.*/
7:     $O_{t+1} \leftarrow$ ES($O_t \bigcup Q$); /*Environmental selection.*/
8:     $P_{t+1} \leftarrow$ AMP($t$,$L$,$P_t$); /*Get $P_{t+1}$.*/
9:     $A \leftarrow$ UNS($A$); /*Update $A$.*/
10:    $t$++;
11: end while
12: Output $A$.
```

4.2. Initialize population

The initial population is composed of N individuals. According to the characteristics of BSPs, an encoding method based on job permutation is adopted, where each individual represents a scheduling scheme. There are two steps to generate the initial population. First, three special individuals are generated through three heuristics, i.e., the jobs are sorted according to the rules of Longest Processing Time (LPT), Shortest Processing Time (SPT), and Earliest Release Time (ERT), respectively. Second, the other individuals of the initial population are generated randomly.

Before being evaluated, each individual needs to be decoded. The heuristic of decoding is described as follows.
Step 1: For each job in the input job permutation, assign this job to a batch so that the constraint of machine capacity is satisfied. If there is no any batch that can accommodate this job, create a new empty batch and assign this job into this new batch.
Step 2: Sort the batches in ascending order of their release time.
Step 3: For each batch in the sequence of batches, assign it to the earliest available machine. If there is more than one machine, choose the machine with the faster processing speed.
Step 4: Calculate the objective value of the generated schedule.

4.3. Adaptive clustering

In most clustering algorithms, the number of clusters needs to be specified in advance, which is tough to determine. Moreover, the number of clusters affects the performance of clustering algorithms inevitably. As a result, in this study, an improved adaptive clustering method is designed to extract accurate distribution characteristics of solutions in search space as much as possible. Based on the found distribution characteristics, the number of clusters is able to be adjusted adaptively, instead of being determined beforehand.

As one of the classic partition-based clustering algorithms, K-means [42], facing the difficulty of determining the number of clusters in advance, blindly chooses the initial centroid points. However, K-means has advantages of simple principle, easy implementation, and fast convergence. In other words, K-means can provide a better clustering result for ACBEA based on the exact number of clusters. Moreover, Canopy [43], a kind of "rough clustering" algorithm, though unable to ensure accuracy, can provide an appropriate number of clusters quickly. Based on the above analysis, Canopy+K-means algorithm, an adaptive clustering method, is presented.

In Canopy+K-means algorithm, to eliminate the dependence of K-means algorithm on prior knowledge, Canopy algorithm is generally run to provides reasonable number of clusters and the initial centroid points, firstly. K-means algorithm is then applied to cluster the individuals based on the provided centroid points. However, if this adaptive clustering is used directly in the evolution process, Canopy algorithm has to be frequently executed to determine the number of clusters so as to make K-means algorithm reach convergence after a high number of iterations.

Therefore, the following improvements for Canopy+K-means algorithm are proposed. First, Canopy algorithm is only utilized to redetermine the number of clusters and the initial centroid points every H iterations, instead of being performed at every iteration. Second, the clustering process of K-means algorithm is integrated into the evolutionary process of ACBEA. In each iteration of ACBEA, K-means is used to cluster the population for only one time. As a result, the population are aggregated into several clusters. The details of the improved adaptive clustering method are described as Algorithm 2, where $dis(S_i, S_r)$ denotes the Euclidean distance between S_i and S_r, K denotes the number of clusters, R denotes temporary data set, CY_K denotes Kth canopy in the Canopy algorithm.

In Algorithm 2, if the condition in line 1 is satisfied, Canopy algorithm gives the number of clusters K in lines 5–16, and then the initial centroid points are provided in lines 17–19. In lines 21–23, K-means algorithm partitions the population into K clusters after only one iteration. In lines 24–26, the centroid points of K-means algorithm are updated. In addition, Canopy algorithm needs two distance thresholds T_1, T_2. Hence, based on the sum of Euclidean distances between all individuals in the population, T_2

Algorithm 2 AC(O_t,t,H)

```
Input: Current population ($O_t = \{S_1, \ldots, S_N\}$); current iteration
    $t$; updated iteration $H$.
Output: $Cr$ /*$K$ clusters ($C_1$, ..., $C_k$).*/.
1: if mod($t, H$) $= 0 \parallel t = 1$ then
2:     $R = O_t$;
3:     $K = 0$
4:     Calculate the distance thresholds $T_1, T_2$;
5:     while $|R|$ do
6:         $K$++;
7:         Select $S_r$ from $R$ to initialize $CY_K$;
8:         for $i = 1$ to $|R|$ do
9:             if $dis(S_i, S_r) \le T_1$ then
10:                Add $S_i$ into the $CY_K$;
11:                if $dis(S_i, S_r) \le T_2$ then
12:                    Remove $S_i$ from $R$;
13:                end if
14:            end if
15:        end for
16:    end while
17:    for $k = 1$ to $K$ do
18:        Generate the centroid point $m_k$ using $m_k = \frac{1}{|CY_k|}\sum_{S \in CY_k} S$;
    /*Obtain the initial centroid point $m_k$.*/
19:    end for
20: end if
21: for $i = 1$ to $N$ do
22:    Assign $S_i$ to $C_k$, according $k = \arg\min_{j=1,\ldots,K} dis(S_i, m_j)$;
23: end for
24: for $k = 1$ to $K$ do
25:    $m_k = \frac{1}{|C_k|}\sum_{S \in C_k} S$; /*update the centroid point $m_k$*/
26: end for
27: return $Cr$.
```

is set to the average of the sum of Euclidean distances on all the individuals. The value of T_1 is twice that of T_2.

However, there is an extreme case of using the adaptive clustering method. That is, during the clustering process, the adaptive clustering may divide the entire population into one single cluster, causing that reproduction cannot perform normally. To avoid this, the adaptive clustering method is designed to cluster the entire population into at least two clusters so that the reproduction is applicable.

4.4. Reproduction

After adaptive clustering, the clustering result of K clusters, representing the distribution characteristics of solutions in search space, is applied to the reproduction for the current population. Different from previous studies, each individual selecting the mate randomly, a new recombination restriction strategy (RRS) is presented to meet diversified search demands in this paper. The details of RRS are shown as Algorithm 3. In the RRS, there are two recombination mating modes for each individual, that is, similar mating and dissimilar mating. In lines 1–2, if the probability generated randomly is less than the current mating probability P_t, the similar mating is chosen to select the mate for the current individual; otherwise, the dissimilar mating is chosen, as shown in lines 3–4.

The details of the reproduction are described as Algorithm 4. In line 1, Q, denoting the set of N new individuals, is initialized. In line 3, the appropriate mate S_j is selected for individual S_i according to the RRS. In line 4, two-point crossover is employed in S_i and S_j to generate the offspring r. In line 5, a new individual r' is generated through executing exchange mutation on r. In line 6, r' is added into Q. Finally, a set of N new individuals denoted by Q is constructed according to Algorithm 4.

Algorithm 3 RRS(S_i,P_t,Cr)

```
Input: Current individual $S_i$; mating probability $P_t$; clustering result $Cr$.
Output: Mate $S_j$.
1: if $rand() \leq P_t$ then
2:     Select $S_j$ from the cluster to which $S_i$ belongs; /*Similar mating.*/
3: else
4:     Select $S_j$ from any cluster that does not contain $S_i$; /*Dissimilar mating.*/
5: end if
6: return $S_j$.
```

Algorithm 4 REP(O_t,P_t,Cr)

```
Input: Current population ($O_t = \{S_1, \ldots, S_N\}$); mating probability $P_t$; clustering result $Cr$.
Output: A set of new individuals $Q$.
1: $Q \leftarrow \varnothing$
2: for $i = 1$ to $N$ do
3:     $S_j \leftarrow$ RRS($S_i$,$P_t$,$Cr$);
4:     $r \leftarrow$ Crossover($S_i$,$S_j$);
5:     $r' \leftarrow$ Mutation($r$);
6:     Add $r'$ into $Q$;
7: end for
8: return $Q$.
```

4.5. Adjust mating probability

From the perspective of evolution, in the RRS, the similar mating is able to enhance the local exploitation ability of the algorithm, while the dissimilar mating can improve the global exploration ability of the algorithm. Therefore, the mating probability plays an important role in balancing exploitation and exploration in the proposed algorithm. Moreover, the demands for exploitation and exploration are changeable during the process of evolution. In early stage, global exploration ability of the algorithm needs to be enhanced, while better local exploitation ability is required with the evolution of population. Therefore, to balance between exploitation and exploration, the mating probability is adaptively adjusted in the proposed algorithm. The details of the presented adjustment are described in Algorithm 5.

In Algorithm 5, $value(k)$ denotes the contribution degree of the new individuals generated through similar mating at the kth iteration. $n_1(k)$ denotes the number of new individuals generated through similar mating and selected into the next population Q at the kth iteration, $n_2(k)$ denotes the number of new individuals generated through dissimilar mating and selected into the next population Q at the kth iteration. It is notable that $value(k) \in [0, 1]$. The mating probability is designed to be adaptively adjusted every L iterations. If the adjustment condition is satisfied, the new mating probability is obtained based on the average contribution degree accumulated in the previous L iterations, as described in lines 5–7 and 14, otherwise, the mating probability remains unchanged in lines 2–3. However, there are several special cases to be noted. If the mating probability is too small or too large, the search ability may totally bias toward exploration or exploitation. Therefore, it is necessary to limit the value of mating probability. According to lines 9–10, if the contribution degree accumulated in the previous L iterations is not more than 0.2*L, the mating probability is set to 0.8. If the contribution degree accumulated in the previous L iterations is not less than 0.8*L, the mating probability is set to 0.2, as shown in lines 11–12.

Algorithm 5 AMP(t,L,P_t)

```
Input: Current iteration $t$; historical iteration $L$; mating probability $P_t$.
Output: Mating probability $P_{t+1}$ for the next $t$+1 iteration.
1: $sum$ = 0;
2: if $mod(t, L)! = 0$ then
3:     $P_{t+1} = P_t$;
4: else
5:     for $k = t$-$L$+1 to $t$ do
6:         $value(k) = \frac{n_1(k)}{n_1(k)+n_2(k)}$;
7:         $sum$ += $value(k)$;
8:     end for
9:     if $sum \leq 0.2 * L$ then
10:        $P_{t+1} = 0.8$;
11:    else if $sum \geq 0.8 * L$ then
12:        $P_{t+1} = 0.2$;
13:    else
14:        $P_{t+1} = sum/L$;
15: end if
16: return $P_{t+1}$.
```

4.6. Computational complexity analysis

The complexity of ACBEA is analyzed in this section. Let M denote the number of optimization objectives. At each iteration, four main operations of ACBEA are performed, that is, adaptive clustering, reproduction, environmental selection and update mating probability.

(1) Adaptive clustering is composed of Canopy algorithm and K-means algorithm. The complexities of Canopy algorithm and K-means algorithm are $O(MN^2 + NKM)$ and $O(NKM)$, respectively. Since N is generally far more than K and M, the complexity of adaptive clustering is $O(MN^2)$.

(2) Each individual successively completes mating selection, crossover and mutation in reproduction. Thus, the complexity of reproduction is $O(N)$.

(3) The selection method of NSGA-II is adopted in this paper. Thus, the complexity of environmental selection is $O(MN^2)$.

(4) Contribution degree of L generations is calculated when updating mating probability. Thus, the complexity of updating mating probability is $O(L)$.

Based on the above analysis, adaptive clustering and environmental selection are the key factors that affect the complexity. As a result, the complexity of ACBEA for each iteration is $O(MN^2)$. Therefore, the complexity of ACBEA is $O(TMN^2)$.

5. Computational experiments

5.1. Experimental design

In order to evaluate the overall performance of the proposed algorithm, APMO [34], BODE [21], IMOEA/D [26], and MODDE [18] are served as the comparative algorithms. Additionally, to make the performance of ACBEA more convincing, the proposed strategies of ACBEA are also verified.

The local optimization strategy in [18] is incorporated into each of the compared algorithms to make a fair comparison. Moreover, the methods of encoding and decoding adopted in all the compared algorithms are the same. There are 12 instance

Table 1
Factors and levels of the testing instances.

Factors	Levels	Number of levels
Number of jobs (n)	100, 200, 300	3
Job size (s_j)	U[1, 15], U[15, 35]	2
Job release time (r_j)	U[1, *LB*]	1
Job processing time (p_j)	U[8, 48]	1
Number of machines (m)	3, 5	2
Machine capacity (C)	40	1
Machine processing power (PP_i)	$2v_i^2$	1
Machine standby power (SP_i)	1	1
Machine processing speed (v_i)	1, 1.5, 2, 2.5, 3	5

groups in total according to factors and levels of the instances in Table 1. Ten testing instances are generated randomly for each instance group. Furthermore, each algorithm is run ten times on each testing instance to obtain the final result on this instance.

As shown in Table 1, *LB* is calculated based on the method in [44]. Moreover, there are different processing speeds of machines. For the instances with three (five) machines, the rates of processing speeds of the three (five) machines are set to one, two, and three (one, one point five, two, two point five, and three), respectively.

To calculate the *TEC*, the Beijing time-of-use (TOU) electricity price standard [45] is adopted as the electricity price function, where a day is divided into three time periods and High-peak (July–Sept) period is ignored for simplifying the electricity price function model. The TOU pricing standard is shown as Table 2 and the simplified electricity price function is formulated as Eq. (13).

$$f(t)=\begin{cases}0.4, & 24d-1\le t<24d+7\\ 0.8, & 24d+7\le t<24d+10\\ 1.3, & 24d+10\le t<24d+15\\ 0.8, & 24d+15\le t<24d+18\\ 1.3, & 24d+18\le t<24d+21\\ 0.8, & 24d+21\le t<24d+23\end{cases} \tag{13}$$

where d is a natural number. All the algorithms are programmed in C++ and run on a Windows 10 PC with Core i5, 3.40 GHz processor, and 16GB RAM.

5.2. Parameter settings

There are some parameters that affect the performance of ACBEA. Therefore, it is necessary to determine appropriate values for these parameters. The Taguchi method is one of the popular statistical analysis methods that can obtain better parameter settings through fewer experiments. Therefore, Taguchi is used to determine the parameter values in this paper. The main parameters considered in ACBEA include crossover probability (p_c), mutation probability (p_m), historical iteration (L), and updated iteration (H), which are summarized with three levels in Table 3. According to the Taguchi method, nine instance groups of parameter combinations are designed and listed in Table 4. In the preliminary experiments, there are 12 testing instances, each of which is chosen from one of the 12 instance groups. Moreover, the hypervolume (HV) metric [46] is taken to evaluate the performance of ACBEA with different parameter combinations. The

Table 3
Parameter levels.

Level	p_c	p_m	L	H
1	0.9	0.3	20	20
2	0.7	0.2	15	15
3	0.5	0.1	10	10

Table 4
Orthogonal array for parameter combination pilot experiments.

No	p_c	p_m	L	H
1	0.9	0.3	20	20
2	0.9	0.2	15	15
3	0.9	0.1	10	10
4	0.7	0.3	15	10
5	0.7	0.2	10	20
6	0.7	0.1	20	15
7	0.5	0.3	10	15
8	0.5	0.2	20	10
9	0.5	0.1	15	20

Table 5
Parameter settings.

ACBEA	APMO	BODE	IMOEA/D	MODDE
$N=100$	$N=100$	$N=288$	$N=100$	$N=100$
$T=500$	$T=500$	$T=100/150$	$T=500$	$T=500$
$p_c=0.9$	$CR=1$	$p_m=0.3$	$niche=3$	$W=0.9$
$p_m=0.2$	$F=0.5$	$p_c=0.3$		$F=0.1$
$P_1=0.3$	$\beta=0.5$	$p_s=0.2$		
L=15	$HL=15$			
H=10	$i_1=50/i_2=5$			

average HV value of ACBEA with each parameter combination on the 12 testing instances is used as the response variable. The Taguchi method is performed on Minitab version 19 platform. The final experimental results are shown in Fig. 2. In Fig. 2, the x-axis denotes the levels of parameters, corresponding to Table 3, and the y-axis denotes the average response values for parameters of different levels, denoted by Means. It is noted that the higher the values of Means, the better the results of ACBEA with the combinatorial parameter values. From Fig. 2, it can be observed that ACBEA obtains the best result with $p_c=0.9$, $p_m=0.2$, $L=15$, and $H=10$.

The parameters of all the comparative algorithms are set as Table 5.

As shown in Table 5, for each algorithm except BODE, the population size N and the maximum iteration T are set to 100 and 500, respectively. Due to the particularity of BODE, when the number of jobs is 100 or 200 (300), the maximum iteration is set to 100 (150). p_s denotes the selection probability, which is used for environmental selection. In APMO, CR and F denote the control parameters of DE. β and HL denote the initial mating restriction probability and history length, respectively. Furthermore, the clustering algorithm in APMO is terminated if the current iteration number reaches i_1 or the clustering result keeps unchanged for i_2 iterations. In IMOEA/D, *niche* denotes the neighborhood size of each subproblem. In MODDE, W and F denote the crossover factor and mutation scale factor, respectively. Additionally, the initial mating probability P_1 is set to 0.3 to guarantee better exploration ability of ACBEA at the beginning of search.

Table 2
Beijing's commercial TOU pricing standard.

Type	High-peak (July–Sept)	Peak	Mid-peak	Off-peak
Period	11:00–13:00 20:00–21:00	10:00–15:00 18:00–21:00	7:00–10:00 15:00–18:00 21:00–23:00	23:00–7:00
Price(CNY/kWh)	1.4409	1.3222	0.8395	0.3818

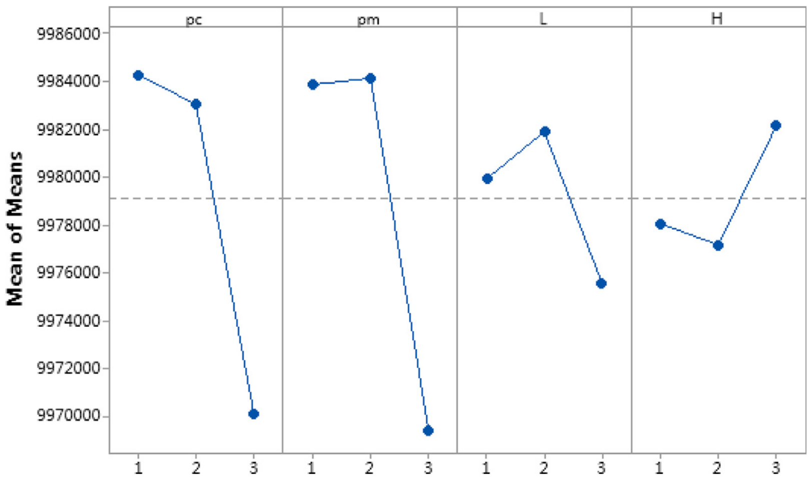

Fig. 2. Results of main effects of Means.

Table 6
Experimental results of adaptive clustering in NNS and HV.

Grp. Code	ACBEA		ACBEA-C		ACBEA-D		ACBEA-K	
	NNS	HV	NNS	HV	NNS	HV	NNS	HV
N1M1S1	**8.8**	2.2701E+07	7.9	**2.2712E+07**	8.3	2.2683E+07	8.5	2.2693E+07
N1M2S1	8.4	2.2415E+07	7.7	2.2419E+07	**9.7**	2.2411E+07	9.0	**2.2420E+07**
N2M1S1	9.0	**1.4310E+07**	8.8	1.4309E+07	**10.2**	1.4302E+07	10.0	1.4292E+07
N2M2S1	6.4	1.3866E+07	6.6	1.3868E+07	**6.9**	**1.3879E+07**	6.4	1.3870E+07
N3M1S1	8.8	**7.6472E+06**	8.1	7.6231E+06	**9.9**	7.6230E+06	9.0	7.6372E+06
N3M2S1	**7.7**	7.2742E+06	6.6	7.2786E+06	6.9	7.2682E+06	6.7	**7.2796E+06**
N1M1S2	14.2	**1.5427E+07**	16.0	1.5421E+07	15.0	1.5415E+07	**16.3**	1.5424E+07
N1M2S2	**10.9**	1.7649E+07	10.7	1.7651E+07	10.2	**1.7653E+07**	10.4	1.7652E+07
N2M1S2	**18.2**	**4.7381E+06**	17.8	4.7286E+06	16.5	4.7300E+06	15.8	4.7227E+06
N2M2S2	8.9	**7.2235E+06**	9.9	7.2120E+06	**11.1**	7.2217E+06	9.4	7.2184E+06
N3M1S2	17.4	2.6099E+05	16.6	**2.6458E+05**	**17.7**	2.6211E+05	17.1	2.5940E+05
N3M2S2	11.7	**1.3376E+06**	11.9	1.3315E+06	11.4	1.3262E+06	**13.1**	1.3286E+06

Table 7
Experimental results of adaptive clustering in *C*.

Grp. Code	*C*(ACBEA, ACBEA-C)	*C*(ACBEA-C, ACBEA)	*C*(ACBEA, ACBEA-D)	*C*(ACBEA-D, ACBEA)	*C*(ACBEA, ACBEA-K)	*C*(ACBEA-K, ACBEA)
N1M1S1	0.305	**0.604**	**0.662**	0.220	**0.522**	0.373
N1M2S1	**0.510**	0.483	**0.533**	0.449	**0.535**	0.482
N2M1S1	**0.622**	0.322	**0.702**	0.248	**0.761**	0.175
N2M2S1	**0.459**	0.429	0.380	**0.617**	0.403	**0.492**
N3M1S1	**0.637**	0.371	**0.561**	0.437	**0.502**	0.388
N3M2S1	0.340	**0.633**	0.437	**0.495**	0.454	**0.465**
N1M1S2	**0.476**	0.320	**0.566**	0.313	**0.505**	0.349
N1M2S2	**0.515**	0.370	0.349	**0.600**	0.439	**0.546**
N2M1S2	**0.540**	0.325	**0.544**	0.355	**0.545**	0.322
N2M2S2	**0.595**	0.354	**0.549**	0.376	**0.516**	0.372
N3M1S2	0.324	**0.559**	0.467	**0.492**	**0.516**	0.416
N3M2S2	**0.630**	0.309	**0.757**	0.240	**0.638**	0.277

Table 8
Experimental results of recombination restriction in NNS, *C*, and HV.

Grp. Code	ACBEA			ACBEA-NRS		
	NNS	*C*(ACBEA, ACBEA-NRS)	HV	NNS	*C*(ACBEA-NRS, ACBEA)	HV
N1M1S1	**8.8**	**1**	**2.3500E+07**	7.4	0	2.3104E+07
N1M2S1	**8.4**	**1**	**2.3209E+07**	7.7	0	2.2923E+07
N2M1S1	**9.0**	**1**	**1.4951E+07**	7.9	0	1.4376E+07
N2M2S1	6.4	**1**	**1.4496E+07**	**6.5**	0	1.4126E+07
N3M1S1	8.8	**1**	**8.1421E+06**	**9.1**	0	7.6584E+06
N3M2S1	7.7	**1**	**7.7524E+06**	**8.4**	0	7.4382E+06
N1M1S2	**14.2**	**1**	**1.6090E+07**	7.9	0	1.5852E+07
N1M2S2	**10.9**	**1**	**1.8361E+07**	8.4	0	1.8116E+07
N2M1S2	**18.2**	**1**	**5.1086E+06**	7.4	0	4.8088E+06
N2M2S2	**8.9**	**1**	**7.6873E+06**	7.0	0	7.3022E+06
N3M1S2	**17.4**	**1**	**3.5066E+05**	6.8	0	2.5078E+05
N3M2S2	**11.7**	**1**	**1.5594E+06**	8.3	0	1.2437E+06

Table 9
Experimental results of adaptive adjustment in NNS and HV.

Grp. Code	ACBEA		ACBEA-P1		ACBEA-P2		ACBEA-P3	
	NNS	HV	NNS	HV	NNS	HV	NNS	HV
N1M1S1	8.8	**2.3879E+07**	7.6	2.3484E+07	9.5	2.3465E+07	**9.6**	2.3461E+07
N1M2S1	**8.4**	**2.3583E+07**	7.2	2.3285E+07	7.0	2.3288E+07	7.2	2.3290E+07
N2M1S1	9.0	**1.5296E+07**	**9.4**	1.4717E+07	7.8	1.4703E+07	8.2	1.4705E+07
N2M2S1	6.4	**1.4830E+07**	**7.6**	1.4447E+07	5.6	1.4454E+07	6.1	1.4442E+07
N3M1S1	8.8	**8.4553E+06**	10.0	7.9772E+06	8.4	7.9460E+06	**10.5**	7.9615E+06
N3M2S1	**7.7**	**8.0493E+06**	7.3	7.7221E+06	7.0	7.7373E+06	6.8	7.7236E+06
N1M1S2	**14.2**	**1.6376E+07**	8.3	1.6125E+07	8.7	1.6129E+07	6.8	1.6135E+07
N1M2S2	**10.9**	**1.8656E+07**	7.8	1.8398E+07	9.2	1.8408E+07	8.9	1.8416E+07
N2M1S2	**18.2**	**5.2677E+06**	9.2	4.9566E+06	8.7	4.9621E+06	7.3	4.9628E+06
N2M2S2	**8.9**	**7.8651E+06**	7.8	7.4707E+06	7.9	7.4751E+06	7.0	7.4799E+06
N3M1S2	**17.4**	**3.8714E+05**	6.7	2.8166E+05	5.9	2.8196E+05	6.0	2.8240E+05
N3M2S2	**11.7**	**1.6223E+06**	9.1	1.2984E+06	8.3	1.3143E+06	9.1	1.3207E+06

Table 10
Experimental results of adaptive adjustment in C.

Grp. Code	C(ACBEA, ACBEA-P1)	C(ACBEA-P1, ACBEA)	C(ACBEA, ACBEA-P2)	C(ACBEA-P2, ACBEA)	C(ACBEA, ACBEA-P3)	C(ACBEA-P3, ACBEA)
N1M1S1	**1**	0	**1**	0	**1**	0
N1M2S1	**1**	0	**1**	0	**1**	0
N2M1S1	**1**	0	**1**	0	**1**	0
N2M2S1	**1**	0	**1**	0	**1**	0
N3M1S1	**1**	0	**1**	0	**1**	0
N3M2S1	**1**	0	**1**	0	**1**	0
N1M1S2	**1**	0	**1**	0	**1**	0
N1M2S2	**1**	0	**1**	0	**1**	0
N2M1S2	**1**	0	**1**	0	**1**	0
N2M2S2	**1**	0	**1**	0	**1**	0
N3M1S2	**1**	0	**1**	0	**1**	0
N3M2S2	**1**	0	**1**	0	**1**	0

5.3. Performance metrics

To measure the performance of the proposed algorithm, three metrics are adopted, namely, the number of non-dominated solutions, the coverage [46] and the hypervolume. The details of these metrics are given as follows.

(1) Number of non-dominated solutions (NNS): This metric indicates the number of non-dominated solutions obtained by each algorithm. The more the non-dominated solutions obtained, the more choices can be provided for decision makers, which is preferable.

(2) Coverage (C) : This metric is used to compare the coverage degree between two non-dominated solution sets, denoted by X and Y, respectively. $C(X, Y)$ denotes the percentage of solutions in Y dominated by or equal to at least one solution in X. $C(X, Y)$ can be calculated as follows.

$$C(X,Y)=\frac{\mid \{y\in Y|\exists x\in X : x\succ y \; or \; x=y\}\mid}{\mid Y\mid} \tag{14}$$

where $x \succ y$ denotes solution x dominates solution y. The value of $C(X, Y)$ is between 0 and 1. $C(X, Y) > C(Y, X)$ means that set X outperforms set Y in terms of dominance relation.

(3) Hypervolume (HV): This metric is used to evaluate the approximation degree between one non-dominated solution set obtained by the algorithm and the true Pareto front. The larger the value of HV, the closer the non-dominated solution set to the true Pareto front, indicating better convergence and diversity of the algorithm. In this paper, $(Z_{C_{\max}}, Z_{TEC})$ is chosen as the reference point for calculating HV, where $Z_{C_{\max}}$ and Z_{TEC} denote the maximum $C_{\max}$ and the maximum TEC among all the non-dominated solutions found by ACBEA and the comparative algorithms on all testing instances, respectively.

5.4. Experimental results

The proposed strategies in ACBEA include: (1) improved adaptive clustering method; (2) new recombination restriction strategy; (3) adaptive adjustment of mating probability. The performance of the proposed algorithm is verified in two steps. First, the effectiveness of the adopted strategies are discussed. Second, the overall performance of all the comparative algorithms is analyzed. The comparative results are shown in Tables 6–12, where the symbols in Column 1 of each table, i.e., “Grp.Code”, denote the codes of instance groups of different levels based on Table 1. For example, “N2M1S2” denotes the group of ten instances with 200 large-sized jobs on three machines, and “N3M2S1” denotes the group of ten instances with 300 small-sized jobs on five machines. Moreover, for each performance metric, we compute the average results over the ten testing instances in the same instance group. Moreover, the best result on each instance group is shown in boldface.

(1) Performance of the improved adaptive clustering method: To demonstrate the effectiveness of the improved adaptive clustering method, three variants of ACBEA are generated. Since Canopy and K-means are adopted in ACBEA, two algorithms of ACBEA with only one of the two methods, i.e., Canopy or K-means, denoted by ACBEA-C and ACBEA-K, respectively, are designed and compared with ACBEA to verify the performance of the combined adaptive clustering strategy in ACBEA. Moreover, DBSCAN [47] is another classic clustering method without determining the number of clusters in advance. Hence, the third comparative algorithm, denoted by ACBEA-D, where the strategy of Canopy+K-means in ACBEA is replaced by DBSCAN, is also developed to compare with ACBEA. In other words, the adaptive clustering method in ACBEA is replaced by algorithms Canopy, K-means, or DBSCAN, respectively. In all, three modified algorithms ACBEA-C, ACBEA-D, and ACBEA-K are compared with ACBEA. The detailed comparative results are listed in Tables 6 and 7.

As shown in Table 6, although worse than ACBEA-D and ACBEA-K on eight instance groups in terms of NNS metric in

Table 11
Comparative results of the five algorithms using NNS and HV.

Grp. Code	ACBEA		APMO		BODE		IMOEA/D		MODDE	
	NNS	HV	NNS	HV	NNS	HV	NNS	HV	NNS	HV
N1M1S1	8.8	**2.4273E+07**	**10.6**	2.3100E+07	3.9	2.3935E+07	8.2	2.3305E+07	7.2	2.4122E+07
N1M2S1	8.4	**2.3984E+07**	8.2	2.3093E+07	2.8	2.3763E+07	**8.6**	2.329E6+07	6.2	2.3872E+07
N2M1S1	9.0	**1.5475E+07**	10.2	1.3168E+07	4.4	1.5053E+07	9.4	1.3546E+07	**11.2**	1.5011E+07
N2M2S1	6.4	**1.5023E+07**	9.2	1.3513E+07	3.3	1.4795E+07	**9.5**	1.3813E+07	6.5	1.4708E+07
N3M1S1	8.8	**8.4327E+06**	11.0	5.9056E+06	4.4	8.1442E+06	9.7	6.2568E+06	**16.2**	7.9705E+06
N3M2S1	7.7	**8.0479E+06**	**9.7**	6.5755E+06	3.3	7.8371E+06	8.2	6.8071E+06	8.7	7.7608E+06
N1M1S2	**14.2**	**1.6805E+07**	6.9	1.6435E+07	6.1	1.6614E+07	9.1	1.6468E+07	13.9	1.6795E+07
N1M2S2	**10.9**	**1.9156E+07**	9.2	1.8593E+07	4.3	1.8907E+07	8.4	1.8826E+07	9.8	1.9098E+07
N2M1S2	**18.2**	**5.5153E+06**	6.2	4.8551E+06	5.2	5.2768E+06	6.7	4.9405E+06	13.3	5.4069E+06
N2M2S2	8.9	**8.2461E+06**	8.8	7.2345E+06	3.0	7.9411E+06	7.5	7.6838E+06	**9.2**	8.0048E+06
N3M1S2	**17.4**	**4.6324E+05**	5.3	1.8321E+05	4.0	3.8620E+05	7.4	2.3687E+05	9.5	4.0819E+05
N3M2S2	**11.7**	**1.8877E+06**	9.1	9.5756E+05	3.8	1.6678E+06	7.7	1.3597E+06	9.3	1.6224E+06

Table 12
Comparative results of the five algorithms using C.

Grp. Code	C(ACBEA, APMO)	C(APMO, ACBEA)	C(ACBEA, BODE)	C(BODE, ACBEA)	C(ACBEA, IMOEA/D)	C(IMOEA/D, ACBEA)	C(ACBEA, MODDE)	C(MODDE, ACBEA)
N1M1S1	**1**	0	**1**	0	**1**	0	**0.989**	0.008
N1M2S1	**1**	0	**1**	0	**1**	0	**1**	0
N2M1S1	**1**	0	**1**	0	**1**	0	**1**	0
N2M2S1	**1**	0	**1**	0	**1**	0	**1**	0
N3M1S1	**1**	0	**1**	0	**1**	0	**1**	0
N3M2S1	**1**	0	**1**	0	**1**	0	**1**	0
N1M1S2	**1**	0	**1**	0	**1**	0	**0.662**	0.176
N1M2S2	**1**	0	**1**	0	**1**	0	**0.831**	0.150
N2M1S2	**1**	0	**1**	0	**1**	0	**0.988**	0
N2M2S2	**1**	0	**1**	0	**1**	0	**1**	0
N3M1S2	**1**	0	**1**	0	**1**	0	**1**	0
N3M2S2	**1**	0	**1**	0	**1**	0	**1**	0

total, ACBEA outperforms the other algorithms on most instance groups in terms of HV metric, indicating that ACBEA have better convergence and diversity of solutions than the other compared algorithms. In addition, from the values of C in Table 7, it can be observed that ACBEA beats ACBEA-C, ACBEA-D, and ACBEA-K on most instance groups. It demonstrates that the improved adaptive clustering method is beneficial to performance of the algorithm.

(2) Performance of the new recombination restriction strategy: Recombination restriction strategy is actually to make each individual choose the mate according to specified methods. To evaluate the efficiency of the new recombination restriction strategy in ACBEA, a variant of ACBEA without recombination restriction, denoted by ACBEA-NRS, is designed. That is, each individual selects its mate randomly in algorithm ACBEA-NRS.

From the experimental results of NNS metric in Table 8, ACBEA-NRS only outperforms ACBEA on three instance groups, while ACBEA can obtain more non-dominated solutions on the other nine instance groups. Furthermore, it can be found that the advantage of ACBEA becomes more obvious on all instance groups with large-sized jobs. Moreover, the values of C(ACBEA,ACBEA-NRS) are all one and the values of C(ACBEA-NRS,ACBEA) are all zero, indicating that the solutions obtained by ACBEA completely dominate those obtained by ACBEA-NRS. In other words, ACBEA has better performance than ACBEA-NRS in terms of convergence. In addition, from the values of HV metric, it can be seen that ACBEA is superior to ACBEA-NRS apparently. Based on the above analysis, it indicates that selecting a mate through the proposed method is more effective than randomly choosing a mate. Therefore, the proposed new recombination restriction strategy benefits the performance of the algorithm.

(3) Performance of the adaptive adjustment of mating probability: The adaptive adjustment of mating probability is able to balance between exploitation and exploration in ACBEA. To verify the effectiveness of the adaptive adjustment, the variant of ACBEA with fixed mating probability is chosen for comparison. To make a complete comparison, three different mating probabilities, i.e., 0.2, 0.5, and 0.8, are used in this compared algorithm, respectively. As a result, there are three compared algorithms with different fixed mating probabilities, denoted by ACBEA-P1, ACBEA-P2 and ACBEA-P3, respectively. In other words, the mating probability of each variant of ACBEA is not adjusted. The experimental results are shown as Tables 9 and 10.

According to the experimental results in Table 9, it can be observed that ACBEA obtains less non-dominated solutions than the other three algorithms on instances with small-sized jobs, except “N1M2S1” and “N3M2S1”, while ACBEA can obtain more non-dominated solutions on instances with large-sized jobs. It means that fixed mating probability cannot guarantee the diversity of solutions. In HV metric, ACBEA is significantly better than the three variants of ACBEA on all instances. As can be seen from Table 10, ACBEA also completely covers the three variants of ACBEA. The experimental results demonstrate that the adaptive adjustment of mating probability is able to obtain better convergence and diversity of solutions than the strategy with fixed mating probability. Hence, the adaptive adjustment of mating probability provides a better balance between exploitation and exploration for ACBEA.

(4) Comparison of the overall performance: To evaluate the overall performance of the proposed algorithm on solving the studied multi-objective BSP, algorithms APMO, BODE, IMOEA/D, and MODDE are compared with ACBEA. BODE and MODDE, relatively new methods, have shown better performance in solving scheduling problems similar to the studied problem in this paper. APMO and IMOEA/D is compared to ACBEA in terms of optimization strategy. Similar to ACBEA, APMO is also incorporated with clustering. IMOEA/D designs a dynamic mating strategy and solves a scheduling problem. Based on the above considerations, the four algorithms are selected to compare with ACBEA, and the comparative results are shown in Tables 11 and 12.

As can be seen from Table 11, ACBEA outperforms the other compared algorithms on five instance groups in terms of metric NNS. Moreover, APMO, IMOEA/D and MODDE have advantage

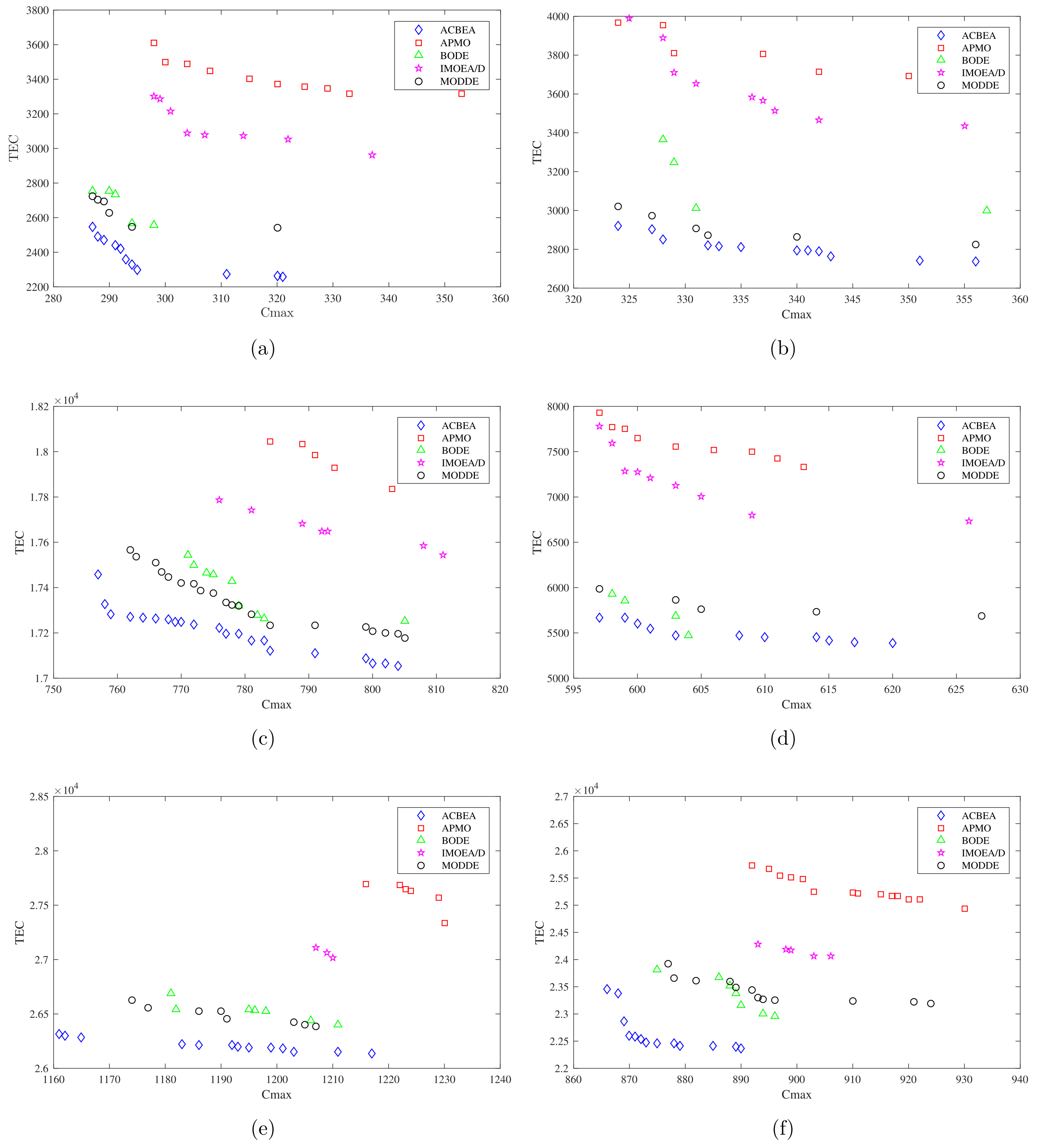

Fig. 3. Pareto fronts found by the five compared algorithm on six testing instances. (a) N1M1S1-6. (b) N1M2S1-8. (c) N2M1S2-3. (d) N2M2S1-9. (e) N3M1S2-7. (f) N3M2S2-8.

in NNS on two or three instance groups, respectively. Although the NNS values of ACBEA are worse than those of the other compared algorithms on seven instance groups, ACBEA beats the other comparative algorithms on all instance groups in terms of metric HV. It shows that ACBEA demonstrates competitive performance on both convergence and diversity of solutions. Furthermore, as shown in Table 12, it is obvious that ACBEA completely covers APMO, BODE and IMOEA/D in terms of metric C. In addition, the values of C(ACBEA,MODDE) are all more than those of C(MODDE,ACBEA) on all instance groups, indicating that ACBEA is able to cover MODDE. In other words, the solutions obtained by ACBEA dominate those of the other comparative algorithms, demonstrating the better convergence of ACBEA.

In addition, to demonstrate the solution quality of the compared algorithms obviously, the non-dominated solutions obtained by the five algorithms on six testing instances are illustrated in Fig. 3. From Fig. 3, it can be seen that ACBEA is able to find better non-dominated solutions than any other compared algorithm in terms of quality and distribution of solutions.

To sum up, it can be concluded that the proposed ACBEA is more effective and efficient in handling the studied multi-objective BSP. That is, both convergence and diversity of solutions obtained by ACBEA are superior to those of any other comparative algorithm.

6. Conclusions

In this paper, we investigate a multi-objective BSP of minimizing makespan and *TEC* simultaneously. To solve this scheduling problem effectively, an adaptive clustering-based evolutionary algorithm, called ACBEA, is devised. In ACBEA, an improved adaptive clustering method is incorporated to extract the distribution characteristics of solutions in search space, which is applied to reproduction. Furthermore, according to distribution characteristics and mating probability, a new recombination restriction strategy is presented to select suitable mate for each individual at the reproduction stage to satisfy different search demands. Finally, the mating probability is adjusted dynamically based on the historical information so that ACBEA can well balance between exploration and exploitation.

The computational experiments are conducted on 12 instance groups which are randomly generated. The effectiveness of the proposed strategies in ACBEA, as well as the performance of ACBEA, is verified. First, the experiments on testing the performance of proposed strategies in ACBEA demonstrates that these strategies are preferable for improving the solution quality of ACBEA. Second, the experimental results of ACBEA compared with other algorithms indicate the efficiency and the effectiveness of ACBEA in addressing the studied multi-objective BSP.

For the future work, clustering results of the multi-generation population can be applied to the algorithm, because it may make the distribution characteristics of solutions more accurate. Besides, objective space and decision space can be integrated as a new similarity measurement of clustering. In addition, other scheduling problems are also worth studying, such as flow shop or job shop with more complex constraints.

CRediT authorship contribution statement

Si-yuan Qian: Methodology, Investigation, Writing - original draft. **Zhao-hong Jia:** Conceptualization, Methodology, Writing - review & editing, Supervision, Funding acquisition. **Kai Li:** Writing - review & editing.

Declaration of competing interest

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

Acknowledgments

This work is supported by the National Natural Science Foundation under grants 71971002, 71601001 and 61872002, the Humanities and Social Science Fund of Ministry of Education of China under grant 15YJC630041, the Natural Science Foundation of Anhui Provincial Department of Education under grant KJ2015A062.

References

[1] G. Ismayilov, H.R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing, Future Gener. Comput. Syst. 102 (2020) 307–322.
[2] X. Ye, S. Liu, Y. Yin, Y. Jin, User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm, Knowl.-Based Syst. 135 (2017) 113–124.
[3] T. Zhu, T. Shi, J. Li, Z. Cai, X. Zhou, Task scheduling in deadline-aware mobile edge computing systems, IEEE Internet Things J. 6 (3) (2019) 4854–4866.
[4] Y. Hu, H. Zhou, C.D. Laat, Z. Zhao, Concurrent container scheduling on heterogeneous clusters with multi-resource constraints, Future Gener. Comput. Syst. 102 (2020) 562–573.
[5] Q. Chen, Q. Pan, B. Zhang, J. Ding, J. Li, Effective hot rolling batch scheduling algorithms in compact strip production, IEEE Trans. Autom. Sci. Eng. 16 (4) (2019) 1933–1951.
[6] J. Wang, F. Qiao, F. Zhao, J.W. Sutherland, Batch scheduling for minimal energy consumption and tardiness under uncertainties: A heat treatment application, CIRP Ann.-Manuf. Technol. 65 (1) (2016) 17–20.
[7] S. Wang, M. Liu, F. Chu, C. Chu, Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration, J. Cleaner Prod. 137 (2016) 1205–1215.
[8] Z. Zeng, M. Hong, Y Man, J. Li, Y. Zhang, H. Liu, Multi-object optimization of flexible flow shop scheduling with batch process-consideration total electricity consumption and material wastage, J. Cleaner Prod. 183 (2018) 925–939.
[9] Y. Ikura, M. Gimple, Efficient scheduling algorithms for a single batch processing machine, Oper. Res. Lett. 5 (2) (1986) 61–65.
[10] R. Uzsoy, Scheduling a single batch processing machine with non-identical job sizes, Int. J. Prod. Res. 32 (7) (1994) 1615–1635.
[11] C.-Y. Lee, R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals, Int. J. Prod. Res. 37 (1) (1999) 219–236.
[12] C.N. Potts, M.Y. Kovalyov, Scheduling with batching: A review, European J. Oper. Res. 120 (2) (2000) 228–249.
[13] M. Mathirajan, A.I. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, Int. J. Adv. Manuf. Technol. 29 (9–10) (2006) 990–1001.
[14] J. Cheng, F. Chu, C. Chu, Bi-objective optimization of single-machine batch scheduling under time-of-use electricity prices, RAIRO-Oper. Res. 50 (4–5) (2016) 715–732.
[15] J. Cheng, F. Chu, M. Liu, P. Wu, W. Xia, Bi-criteria single-machine batch scheduling with machine on/off switching under time-of-use tariffs, Comput. Ind. Eng. 112 (2017) 721–734.
[16] S. Zhang, A. Che, X. Wu, C. Chu, Improved mixed-integer linear programming model and heuristics for bi-objective single-machine batch scheduling with energy cost consideration, Eng. Optim. 50 (8) (2018) 1380–1394.
[17] Z. Jia, Y. Zhang, J.Y.-T. Leung, K. Li, Bi-criteria ant colony optimization algorithm for minimizing makespan and energy consumption on parallel batch machines, Appl. Soft Comput. 55 (2017) 226–237.
[18] S. Zhou, X. Li, N. Du, Y. Pang, H. Chen, A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost, Comput. Oper. Res. 96 (2018) 56–68.
[19] M. Tan, H. Yang, Y. Su, Genetic algorithms with greedy strategy for green batch scheduling on non-identical parallel machines, Memetic Comput. 11 (2019) 439–452.
[20] X. Zheng, S. Zhou, R. Xu, H. Chen, Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm, Int. J. Prod. Res. 12 (2019) 1–18.
[21] M. Liu, X. Yang, F. Chu, J. Zhang, C. Chu, Energy-oriented bi-objective optimization for the tempered glass scheduling, Omega 90 (2020) 1–21.
[22] C. Liu, Approximate trade-off between minimisation of total weighted tardiness and minimisation of carbon dioxide (CO_2) emissions in bi-criteria batch scheduling problem, Int. J. Comput. Integr. Manuf. 27 (8) (2014) 759–771.
[23] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[24] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, 2001.
[25] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[26] E. Jiang, L. Wang, An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time, Int. J. Prod. Res. 57 (6) (2019) 1756–1771.

[27] E. Zitzler, S. Kunzli, Indicator-based selection in multiobjective search, in: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, 2004, pp. 832–842.
[28] J. Bader, E. Zitzler, HypE: an algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (1) (2011) 45–76.
[29] M. Gong, L. Jiao, G. Cheng, C. Liu, Clustering-based selection for evolutionary multi-objective optimization, in: Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009, pp. 255–259.
[30] S.R. Spea, A.A.A. EL Ela, M.A. Abido, Multi-objective differential evolution algorithm for environmental-economic power dispatch problem, in: Proceedings of the 2010 IEEE International Energy Conference and Exhibition, 2010, pp. 841–846.
[31] Y. Hua, Y. Jin, K. Hao, A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular pareto fronts, IEEE Trans. Cybern. 49 (7) (2019) 2758–2770.
[32] Y. Wang, C. Chen, Y. Tao, Z. Wen, B. Chen, H. Zhang, A many-objective optimization of industrial environmental management using NSGA-III: A case of China's iron and steel industry, Appl. Energy 242 (2019) 46–56.
[33] M. Kotinis, Improving a multi-objective differential evolution optimizer using fuzzy adaptation and k-medoids clustering, Soft Comput. 18 (4) (2014) 757–771.
[34] H. Zhang, X. Zhang, S. Song, X. Gao, An affinity propagation-based multiobjective evolutionary algorithm for selecting optimal aiming points of missiles, Soft Comput. 21 (11) (2017) 3013–3031.
[35] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive multiobjective optimization by MOEA/D with Gaussian process model, IEEE Trans. Evol. Comput. 14 (3) (2010) 456–474.
[36] J. Sun, H. Zhang, A. Zhou, Q. Zhang, K. Zhang, A new learning-based adaptive multi-objective evolutionary algorithm, Swarm Evol. Comput. 44 (2019) 304–319.
[37] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering based evolutionary algorithm for large-scale many-objective optimization, IEEE Trans. Evol. Comput. 22 (1) (2018) 97–112.
[38] Z. Wang, Z. Zhan, Y. Lin, W. Yu, H. Yuan, T. Gu, S. Kwong, J. Zhang, Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems, IEEE Trans. Evol. Comput. 2 (6) (2019) 894–908.
[39] Y.-S. Lee, S.-B. Cho, Solving graph coloring problem by fuzzy clustering-based genetic algorithm, in: Simulated Evolution and Learning, Springer Berlin Heidelberg, 2012, pp. 351–360.
[40] L. Cai, S. Qu, Y. Yuan, X. Yao, A clustering-ranking method for many-objective optimization, Appl. Soft Comput. 35 (C) (2015) 681–694.
[41] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1) (1979) 287–326.
[42] J. Macqueen, Some methods for classification and analysis of multiVariate observations, in: Proceedings of Berkeley Symposium on Mathematical Statistics and Probability, 1965.
[43] A. McCallum, K. Nigam, L.H. Ungar, Efficient clustering of high-Dimensional data sets with application to reference matching, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 169–178.
[44] P. Damodaran, M.C. Velez-Gallego, Heuristics for makespan minimization on parallel batch processing machines with unequal job ready times, Int. J. Adv. Manuf. Technol. 49 (9–12) (2010) 1119–1128.
[45] J. Ding, S. Song, R. Zhang, R. Chiong, C. Wu, Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches, IEEE Trans. Autom. Sci. Eng. 13 (2) (2016) 1138–1154.
[46] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
[47] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

Siyuan Qian is currently working toward his M.S. degree in Computer Science and Technology in Anhui University. His research interest includes intelligent optimization.

Zhaohong Jia is a Professor in Anhui University. She received her M.S. degree in Computer Science in 2003 from Anhui University and her Ph.D. degree in Management Science and Technology in 2008 from University of Science and Technology of China. Her research interests include intelligent computation and its applications, scheduling and operations research.

Kai Li is a Professor in Hefei University of Technology. He received his M.S. and Ph.D. degrees from Hefei University of Technology, in 2005 and 2009, respectively. His research interests include operations research, manufacturing system optimization and service system optimization.