Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2015, Article ID 147980, 13 pages
http://dx.doi.org/10.1155/2015/147980

Research Article

A Novel Algorithm to Scheduling Optimization of Melting-Casting Process in Copper Alloy Strip Production

Xiaohui Yan, Zhicong Zhang, Jianwen Guo, Shuai Li, and Kaishun Hu

Department of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China

Correspondence should be addressed to Xiaohui Yan; yxhsunshine@gmail.com

Received 30 March 2015; Accepted 30 May 2015

Academic Editor: Felix T. S. Chan

Melting-casting is the first process in copper alloy strip production. The schedule scheme on this process affects the subsequent processes greatly. In this paper, we build the multiobjective model of melting-casting scheduling problem, which considers minimizing the makespan and total weighted earliness and tardiness penalties comprehensively. A novel algorithm, which we called Multiobjective Artificial Bee Colony/Decomposition (MOABC/D) algorithm, is proposed to solve this model. The algorithm combines the framework of Multiobjective Evolutionary Algorithm/Decomposition (MOEA/D) and the neighborhood search strategy of Artificial Bee Colony algorithm. The results on instances show that the proposed MOABC/D algorithm outperforms the other two comparison algorithms both on the distributions of the Pareto front and the priority in the optimal selection results.

1. Introduction

Copper strip production belongs to nonferrous metals industry. And it is a kind of mixed-production processing. In a typical copper strip company, it mainly includes melting-casting plant and rolling plant. In melting-casting plant, pure cooper ingot and specific small amounts of elements are melted into specific cooper alloy and then casted into alloy ingots. The manufacturing model of this part is flow-processing. The processes in rolling plant are more than in melting-casting plant. It mainly includes hot rolling, two-side milling, bloom rolling, side cutting, annealing, and finished rolling, as seen in Figure 1 [1]. Processes before bloom rolling for different products are the same, so the ingots are processed as a whole. On bloom rolling mill, copper ingots will be rolled to thin coils, thicknesses of which are only 7–15 millimeters. After that, different products may have different technological processes, so the ingots may be cut into different coils according to the orders and then processed as their own technological processes. Manufacturing model on some processes such as heating, annealing, and washing are flow-processing and the rest are discrete processing.

As the first process, melting-casting is important to the whole production. The producing sequence of melting-casting determines the work in process and its processing sequence in rolling part in a large degree. Usually, the copper alloy enterprises calculate the casting dates according to delivery dates and experiential processing cycle after receiving orders. Then, the ingots demanding plan is determined by grouping these orders while considering their alloy kinds, harnesses, technological processes, and casting dates [2]. Different from steel industry, orders of copper alloy strip enterprises are always in small amount, sudden, and with different alloy kinds. As a result, how to optimize the scheduling of melting-casting is very important to the copper alloy strip production. This is what we focus on in this paper.

In the melting-casting process, pure cooper ingots are melted in the smelting furnace. And then, one or more specific small amounts of elements, such as beryllium, silicon, nickel, tin, zinc, chromium, and silver, are added into the molten copper. If the ingredient matches the requirements of the specific cooper alloy kind, the molten cooper alloy is poured into heat holding furnace and finally poured into casting mold from heat holding furnace to form alloy ingots. What should be mentioned is that smelting furnaces used in copper alloy enterprises are mostly electric induction furnaces (the schematic diagram of electric induction furnaces are shown in Figure 2) [3]. When pouring molten cooper alloy liquid into heat holding furnace, not all the liquid is poured. Leaving

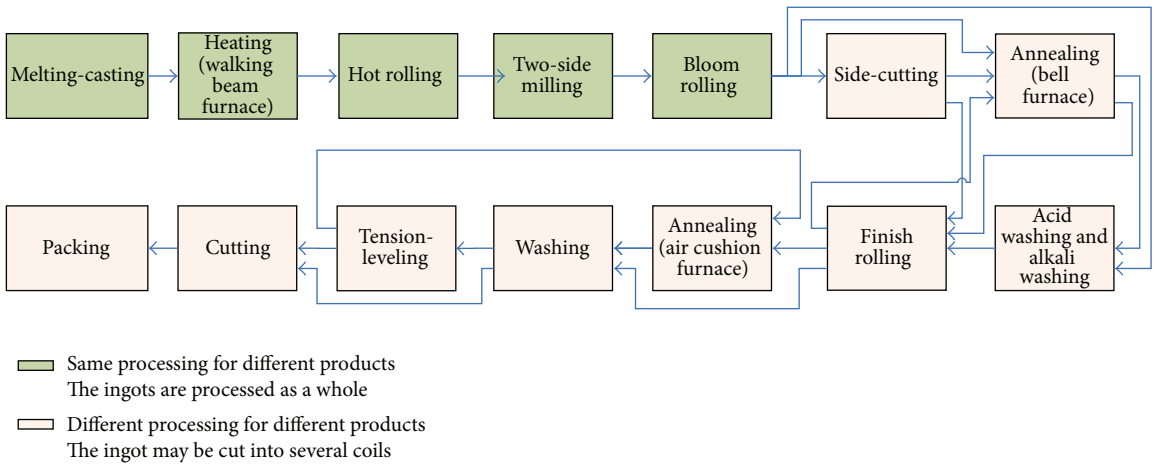
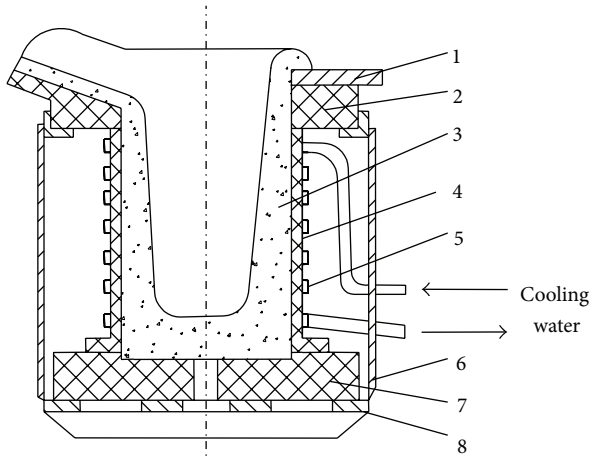


Figure 1: Technological process for copper alloy strip production.



(1) Cover with cement and asbestos; (2) upper frame with firebricks; (3) crucible; (4) insulation cloth; (5) induction coil; (6) protection plate; (7) pedestal with firebricks; (8) Aluminum frame

Figure 2: Schematic diagram of electric induction furnace.

part of the liquid (usually quarter to one-third) in the smelting furnace is needed, which is called "remaining melt." The remaining melt can make the next melting process faster and protect the wall of the furnace. As the main constituents and impurity elements of different alloy kinds are different from each other (maybe differ by several to dozens of times for different alloy kinds), and restrictions for impurity element are strict, melting for different alloy kinds may not be changed directly. If we change the production from an alloy kind with high content of specific impurity element to another alloy kind with lower content of the same impurity element, this requires the addition of pure copper to dilute the remaining melt in order to make the impurity element in accord with the standard. This process is called "diluting remaining melt." The copper melt for diluting cannot produce useful ingot. It will be casted to scrapped ingots using a special vessel. Diluting remaining melt frequently will waste large amount of electric energy and occupy cooper material extremely. And it will prolong the producing period. We should void it as much as possible.

When we make the melting scheduling, we should produce the products not late but also not too early to ensure delivery on time and avoid the backlog of inventory. However, we should also consider economic lot size to guarantee the smoothness of production. At the same time, it could reduce the times of diluting remaining melt and shorten total production cycle. In general, these two objectives are conflicting with each other. The economic lot size principle demands that orders with the same alloy kind are melted and casted in centralized periods, and orders with different alloy kinds are melted in the sequence of content of impurity element increasing gradually. However, this principle could lead to the delay in delivery for parts of orders.

Currently, researches of production scheduling for metallurgy industry mainly focus on steel industry [4–7]. Compared with steel industry, processes of copper alloy strip producing industry are more complex, and the batch of orders and work in process are smaller. All these conditions make the scheduling optimization of copper alloy production more difficult. As a result, there is few related researches in this area. In this paper, we build the model of scheduling optimization of melting-casting process in copper alloy strip production, which considered minimizing both total completion time (makespan) and total weighted earliness and tardiness penalties. And a new algorithm, which we called Multiobjective Artificial Bee Colony/Decomposition (MOABC/D), is employed to solve the model.

2. Multiobjective Model of Scheduling Optimization of Cooper Melting-Casting

The optimization problem for cooper alloy melting-casting scheduling can be seen as a specific single-machine scheduling model. N jobs will be processed on one machine, and the goal is finding a process sequence to optimize some

production target [8]. The difference and specificity lies in the following:

(1) Melting process belongs to flow-processing. The process time for an order depends on the lot size of the order.

(2) Because diluting remaining melt for melting different alloy kinds may be needed, the preparation time of each melting task is not independent. It is related to the previous task in the scheduling sequence.

(3) There is no penalty for earliness in many scheduling models. However, to avoid delay of delivery and backlog of inventory, we will set penalties both if the products are completed late or completed too early to the delivery date.

It has been mentioned above that the orders will be preprocessed after being received. Schedulers will calculate the casting dates for each order and group them to form the ingots demanding plan. Without loss of generality, we suppose that the orders below are all preprocessed and given in the format of ingots demanding plan. In other words, each order will contain the information of alloy kind, quantity of ingots demanded, and estimated melting date. Delivery on time is guaranteed by melting on time.

Based on the descriptions above, we build the model of the melting-casting scheduling problem. For a scheduling instance with n orders, we suppose there is a scheduling solution $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$; π_i represents the sequence number of these orders; then the completion time of each order and total completion time (makespan) are as follows:

$$\begin{aligned} C(\pi_1) &= P(\pi_1), \\ C(\pi_i) &= C(\pi_{i-1}) + S(\pi_{i-1}, \pi_i) + P(\pi_i), \\ & \qquad i = 2, \ldots, n, \\ C_{\max}(\pi) &= C(\pi_n). \end{aligned} \tag{1}$$

$C(\pi_i)$ is the completion time for order π_i and $C_{\max}(\pi)$ is the makespan. $P(\pi_i)$ represents the process time for order π_i. $S(\pi_{i-1}, \pi_i)$ is the potential time for diluting remaining melt from producing order π_{i-1} to order π_i. It is determined by the alloy kinds of the two orders.

The weighted earliness and tardiness penalties for each order π_i and total penalties for solution π are

$$\begin{aligned} L(\pi_i) &= \omega(\pi_i) \left| C(\pi_i) - D(\pi_i) \right|, \\ L(\pi) &= \sum_{i=1}^{n} L(\pi_i), \end{aligned} \tag{2}$$

where $D(\pi_i)$ is the theoretical melting date for order π_i and $\omega(\pi_i)$ is penalty coefficient. It has to be mentioned that both tardiness and earliness of completion will be penalized and calculated in $L(\pi)$.

According to the analysis above, we focus mainly on delivery on time and reducing the times of diluting remaining melt. So we set two objectives in our model: minimizing the total earliness and tardiness penalties and minimizing the makespan. Consider

$$\begin{aligned} &\min f_1 = L(\pi), \\ &\min f_2 = C_{\max}(\pi). \end{aligned} \tag{3}$$

3. Multiobjective Optimization and MOABC/D Algorithm

3.1. Multiobjective Optimization. In the last section, we have built the multiobjective model of the scheduling problem. Multiobjective optimization is a kind of problems that people encountered frequently whether in theoretical research or in practical application [9–11]. It can be stated as finding a solution x which minimizes the objective function $F(X)$ with conflicting objects, as seen in

$$\begin{aligned} &\text{minimize } F(x) \\ &\quad = \left(f_1(x), f_2(x), f_3(x), \ldots, f_m(x)\right)^T, \quad x \in X, \end{aligned} \tag{4}$$

where $x \in X$, solution x is decision vector, and X is the decision space. $y = (f_1(x), f_2(x), \ldots, f_m(x)) \in Y$ and Y is the objective space. Unlike single-objective optimization, solutions of multiobjective problems are in such a way that the performance of each objective cannot be improved without sacrificing performance of at least another one. Hence, the solution to a multiobjective optimization (MOO) problem exists in the form of an alternate tradeoff known as Pareto set. The Pareto set is defined based on Pareto dominance. Let x^1 and x^2 be two decision vectors in solution space X; x^1 is said to dominate x^2 (denoted by $x^1 \prec x^2$), if $f_i(x^1) \leq f_i(x^2)$ for all $i = 1, \ldots, n$, and $x^1 \neq x^2$. A solution $x^* \in X$ is called Pareto optimal if there is no $x \in X$ such that $F(x) \prec F(x^*)$. The set of all Pareto optimal is called the Pareto set, denoted by PS. The set of all Pareto objective vectors, PF $= \{y \in Y \mid y = F(x), x \in \text{PS}\}$, is called the Pareto front of the multiobjective problem [12].

In the last decades, scholars proposed many approaches to solve it. At first, researchers use method of weighting, analytic hierarchy process, and so on to transform the multiobjective problems to single-objective problems [13, 14]. These methods are simple and convenient. However, the weights and values are always determined by subjective experiences and lack of scientific basis. After that, various algorithms which are based on multiobjective decision theory are proposed. The first generation of multiobjective algorithms mostly adopts the rules of Pareto dominant and fitness assignment, such as Multiobjective Genetic Algorithm (MOGA) [15] and Nondominated Sorting Genetic Algorithm (NSGA) [16]. The characteristic of second generation of multiobjective algorithms is employing elitist individual preserving strategy, such as Pareto Envelope-Based Selection Algorithm (PESA) [17], Microgenetic Algorithm (Micro-GA) [18], and NSGA-II (the improved version of NSGA) [19]. NSGA-II algorithm is widely used in many engineering fields due to its well performance.

In 2006, professor Zhang of Essex proposed Multiobjective Evolutionary Algorithm/Decomposition (MOEA/D)

[20]. Different from the algorithms mentioned above, MOEA/D uses the idea of decomposition. It transforms the multiobjective problem to several single-objective subproblems using a series of weight groups. The Pareto set of the original multiobjective problem is obtained by updating optimal solution of each subproblem. The computation complexity of MOEA/D algorithms is lower than in algorithms mentioned above as it does not need to consider fitness assignment and crowding distances. Experiments on benchmark functions show that MOEA/D is superior to NSGA-II both on the convergence and uniformity of distribution of their Pareto fronts [20].

As a result, we set out to use the idea of MOEA/D algorithm to solve our multiobjective problem. However, melting-casting scheduling optimization is a kind of discrete problem. It has its own characteristics and is different from the test benchmark functions used in most studies. To apply the MOEA/D algorithm on melting-casting scheduling optimization, we need to add some extra methods to adapt it to the practical problem. Besides, studies on scheduling optimization indicate that neighborhood search is important for scheduling optimization [21], which provides us with inspiration. Artificial Bee Colony (ABC) is a novel swarm intelligence algorithm proposed by Karaboga and Basturk recently [22]. In this algorithm, employed bees and onlooker bees use the idea of neighborhood search to find best nectar source. ABC algorithm is proved to perform well on many optimization instances [23, 24]. In this paper, we combined the framework of MOEA/D and the neighborhood search of ABC and proposed a new approach, which is named MOABC/D. With the idea of decomposition and the optimization ability of ABC, we hope that the MOABC/D algorithm has well performance on solving the melting-casting scheduling multiobjective problem.

3.2. MOABC/D Algorithm. The framework and steps of MOABC/D are similar to these of MOEA/D. The difference is that neighborhood search was introduced in the new individual producing process. The flowchart of MOABC/D is shown in Figure 3.

Firstly, we generate several groups of weights to decompose the original multiobjective problem into same amount of single-objective subproblems. The weight groups are corresponding with the subproblems one-to-one. The population size of the algorithm is equal to the amount of subproblems too. The ith individual in the population corresponds with a solution of the ith subproblem. Each subproblem has several neighbor subproblems. This is determined by the Euclidean distance of the weights which they corresponded. The kth subproblems whose weights are nearest to the current subproblem's weight become its neighbors. k is a predetermined parameter which is larger than one. So a subproblem will be in its own neighbors set, too. By the decomposition method, we transform obtaining the Pareto set of the multiobjective problem to obtaining the optimal solutions of these subproblems. The fitness of a solution of a subproblem is evaluated by the decomposed distance, which is determined by the weights and distance of the solution to ideal point. The ideal point is the set of optimal values on each objective. In practical problems, the ideal points are mostly unknown. We could use the optimal values we have found in running processes instead. In the iteration, for each individual, we will choose another individual in its neighborhood and use the neighbor search strategy of ABC to produce a new individual. The values on each objective of the new individual are calculated to check if the ideal point needs to be updated. Then, we will try to use the new individual instead of the current solution of the neighbor to see whether the decomposed distance $g(y \mid \lambda, z)$ will reduce. If the decomposed distance reduces, the solution of the neighbor in the population will be replaced by the new individual. The external archive will be updated too. A parameter N_u is used to control the maximum times of trial in order to keep the diversity and escape from premature. The value of N_u equals one-tenth to one-fifth of the amount of neighbors. During the iterations, solutions for each subproblem are updated and improved gradually. And the final external archive composes the Pareto set of the multiobjective problem.

Decomposition is an important feature of MOABC/D and MOEA/D and is also the key to these algorithms. There are several decomposition approaches for converting the problem of approximation of the Pareto front into a number of scalar optimization, such as weighted sum approach, Tchebycheff approach, and boundary intersection approach [25]. In this paper, we use the Tchebycheff approach. The objective of the subproblems is

$$\min g^{\text{te}}(x \mid \lambda, z^*) = \max_{1 \le i \le m} \{\lambda_i \left| f_i(x) - z_i^* \right|\}, \tag{5}$$

where λ is the weight, m is the amount of objectives, z^* is the ideal point, and $z^* = \{f_1^*, f_2^*, \ldots, f_m^*\}^T$, where f_i^* is the optimal value on the ith dimension.

4. MOABC/D for Melting-Casting Scheduling

4.1. Individual Encoding in MOABC/D. The original MOEA/D and ABC algorithms are designed for continuous optimization problems. As the scheduling optimization is discrete optimization problem, we need to redefine the individual encoding rule to make the algorithms adapt with the problem.

As we have mentioned above, the melting-casting scheduling of copper strip producing can be seen as a single-machine scheduling problem. So the solution is a sequence of the orders. For a problem with n orders, the solution could be represented by a vector with n dimensions. There are two typical encoding methods for scheduling problems currently: direct encoding and indirect encoding [26].

With direct encoding method, the individual is represented by a vector with nonoverlapping sequence of integers composed by 1, 2, 3, ... to n. Value on each dimension represented the number of order directly. With indirect encoding method, the individual is represented by an arbitrary vector whose dimension is equal to the amount of the orders. And then the vector is mapped into the solution of the problem according to rules such as ranked-order-value (ROV) [27]. In

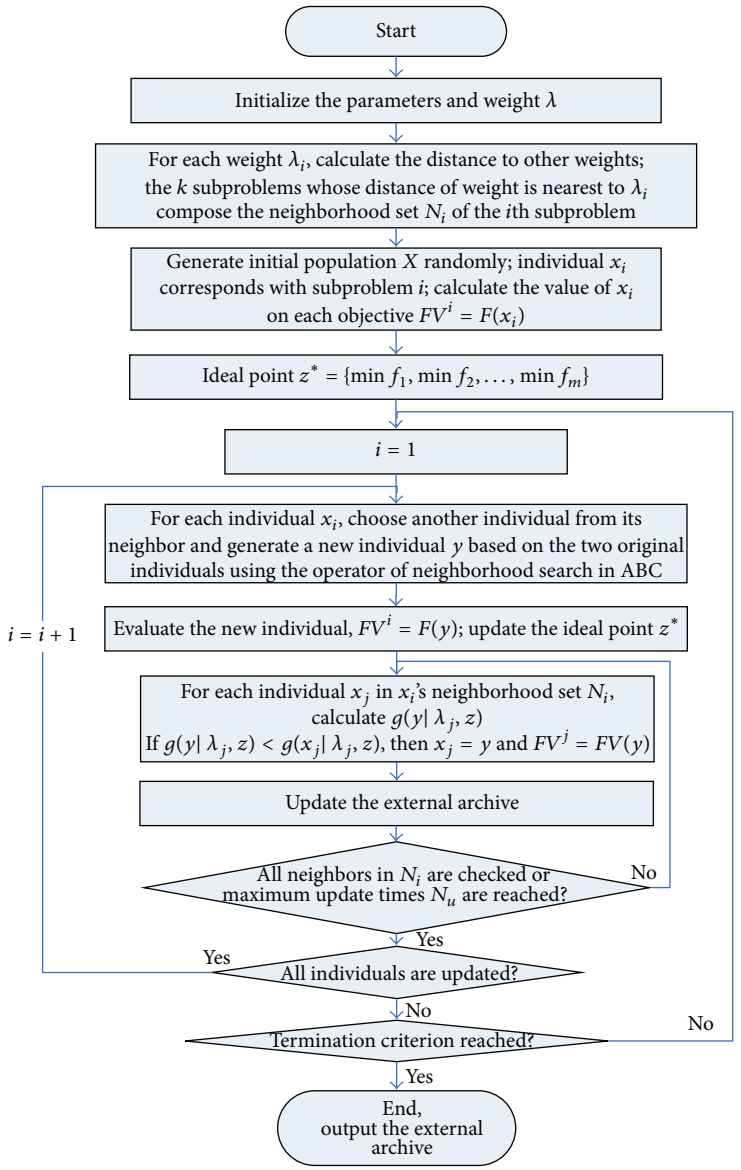


Figure 3: Flowchart of MOABC/D algorithm.

this paper, we used the direct encoding method. The value on each dimension respects the number of order directly.

4.2. New Individual Producing Rules. As the direct encoding method is used, we need to set some rules in the new individual producing process to guarantee that the individuals are always feasible solutions. In MOABC/D, we set three rules. In the new individual producing process, the three rules are employed with equal probabilities to keep the diversity.

Rule 1. For individual x_i, select another individual x_j from its neighbor randomly. Select a dimension p randomly. Find the dimension q in x_i whose value equals x_{ip}. If $q \neq p$, exchange the values of x_i on dimensions p and q to produce the new individual.

Rule 2. For individual x_i, select another individual x_j from its neighbor randomly. Select a dimension p randomly (larger than one). If x_{jp} (the value of x_j on dimension p) $\neq p$, then insert the value of x_i on dimension x_{jp} before dimension p. Otherwise, reselect p and repeat above process.

Rule 3. For individual x_i, select another individual x_j from its neighbor randomly. Select a dimension p randomly. If $x_{jp} \neq p$, then exchange the value of x_i on dimensions p and x_{jp} to produce the new individual. Otherwise, reselect p and repeat above process.

4.3. Select the Best from Pareto Set Based on Fuzzy Set Theory. Multiobjective decision based on Pareto dominance theory believes that there is no superiority between two Pareto

solutions. It could not tell which one is better. However, for practical problems, decision makers have their preferences. For different solutions in Pareto set, they may prefer some solutions than the others. In addition, too many Pareto solutions will reduce the efficiency of decision makers.

Based on the studies of decision makers and their decision-making mechanism, fuzzy set theory makes two hypotheses [28]:

(1) The fuzzy objective can be quantized to membership function by interacting with decision makers.

(2) Membership function is connotative to the decision makers. That is, the decision makers cannot give the exact form of the membership function. But he/she can provide their preference. In addition, the membership function is continuous and increasing.

In our melting-casting scheduling problem, the schedulers must choose one scheduling solution to execute finally. Therefore, only obtaining the Pareto set is not enough. We need to provide the schedulers with the priority of these solutions to help them choose the best ones. Based on investigation to schedulers and their demands, we make two hypotheses to the preference of the schedulers in this problem:

(1) The final solution chosen should balance the two objectives. Solutions which only minimize the makespan or total penalties but perform badly on another objective will not be considered as good choices.

(2) The smaller its distance to the combination of minimum value on each objective (ideal point), the better the solution.

Based on the fuzzy set theory and hypotheses mentioned above, we employed a method to calculate the priority of the solutions in final Pareto set, which can help the schedulers to make their decision.

In the method, we define membership function μ_i^k as the relative fitness for the kth Pareto solution on the ith dimension. And the total relative fitness for the kth Pareto solution is represented by normalized membership function μ^k [29], as seen in

$$\mu_i^k = \begin{cases} 1, & f_i^k = f_i^{\min}, \\ \dfrac{f_i^{\max} - f_i^k}{f_i^{\max} - f_i^{\min}}, & f_i^{\min} < f_i^k < f_i^{\max}, \\ 0, & f_i^k = f_i^{\max}, \end{cases} \tag{6}$$

$$\mu^k = \frac{\sum_{i=1}^{M} \mu_i^k}{\sum_{j=1}^{S} \sum_{i=1}^{M} \mu_i^j}.$$

In (5), $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum values on the ith objective, respectively. M is the amount of objectives. S is the amount of solutions in the Pareto set. The larger the value of μ^k is, the smaller the weighted relative distance of the solution to the ideal point is. In other words, the solution with larger μ^k value could harmonize each objective better and be more favored by the decision makers under the decision preference mentioned above. By sorting the value of μ^k, we can obtain the priority of the Pareto solutions under the preference.

Table 1: Times of diluting remaining melt needed when alloy kind changed.

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	0	0	0	0	0	0	0
A2	1	0	0	0	0	1	0	0
A3	1	0	0	0	0	0	0	0
A4	—	—	—	0	—	—	3	3
A5	2	1	0	0	0	1	0	0
A6	0	0	0	0	0	0	0	0
A7	—	—	—	0	3	—	0	1
A8	—	3	3	0	2	—	0	0

5. Comparison and Discussion of the Results on Testing Instances

In this part, we will test our algorithm on several instances simulated according to practical production in a copper alloy enterprise in Luoyang, China. This enterprise is a leading enterprise in China and also well-known in the whole world in copper alloy industry. Recently, the enterprise launched a cooper alloy strip production line with an annual rate of 10,000,000 kg. The products include oxygen-free copper and IC Frame. On oxygen-free copper furnace, melting is simple as it only melts pure cooper and need not diluting remaining melt. In this paper, we are concerned with the melting-casting scheduling on IC frame furnace only.

The capacity of the IC frame furnace is about 3,600,000 kg per month. The weight of each ingot is 10,000 kg. That is, it can cast 360 ingots per month. Each ingot costs about two hours to melt and cast. On the IC frame furnace, it mainly produces eight kinds of cooper alloy: C10200, LC1100, LCF10, QFe2.5, T2, T2D, TFe0.1, and TP2. The eight alloy kinds are marked as A1–A8, respectively, in Table 1. The times of "diluting remaining melt" needed when alloy kind changed is listed in Table 1 [1].

In Table 1, the first column represents the alloy kind produced before. The first row represents the alloy kind which we want to produce next. The value in the center cells represents the information of diluting remaining melt when the production changed between two alloy kinds. "0" means that the production from the previous alloy kind to the next can be changed directly, and there is no need to diluting remaining melt. "—" means that the content limits of impurity element of the two alloy kinds are too much different. It needs too many times of diluting to make the components meet the requirements. So we cannot change from the previous alloy kind to the targeted alloy kind due to the cost. Other values such as 1, 2, and 3 represent the times needed of diluting remaining melt for changing the alloy kinds. Similar to regular melting-casting, each process of diluting remaining melt costs about two hours too.

In our testing, we simulated three kinds of order data for a month according to the practical demand as follows:

(1) 20 orders: alloy kinds are selected randomly within A1–A8, quantity of ingots demanded are distributed evenly between [1, 30], and estimated melting dates are distributed evenly between [1, 30]

(2) 30 orders: alloy kinds are selected randomly within A1–A8, quantity of ingots demanded are distributed evenly between [1, 20], and estimated melting dates are distributed evenly between [1, 30]

(3) 40 orders: alloy kinds are selected randomly within A1–A8, quantity of ingots demanded are distributed evenly between [1, 18], and estimated melting dates are distributed evenly between [1, 30].

The main difference of the three groups of orders is that their quantity of orders and the average quantity of ingots demanded for each order in three groups are different.

The processing time for each order equals the quantity of ingots demanded divided by the process time for each ingot. Time for diluting remaining melt for each order equals times of diluting needed multiplied by process time for each diluting. As mentioned above, process time for each ingot and process time for each diluting are both two hours. For two orders which cannot be changed directly, we set the times of diluting needed to be 1000 and use the rigorous penalty to eliminate those solutions.

The penalty coefficient $\omega(\pi_i)$ is a piecewise function which related to the tardiness, as seen in

$$\omega(\pi_i) = \begin{cases} 0.2(x+5), & x \le -5 \\ 0, & -5 < x \le 0 \\ 0.5x, & 0 < x \le 5 \\ (x-5)+2.5, & 5 < x \le 10 \\ 2(x-10)+7.5, & x > 10. \end{cases} \quad (7)$$

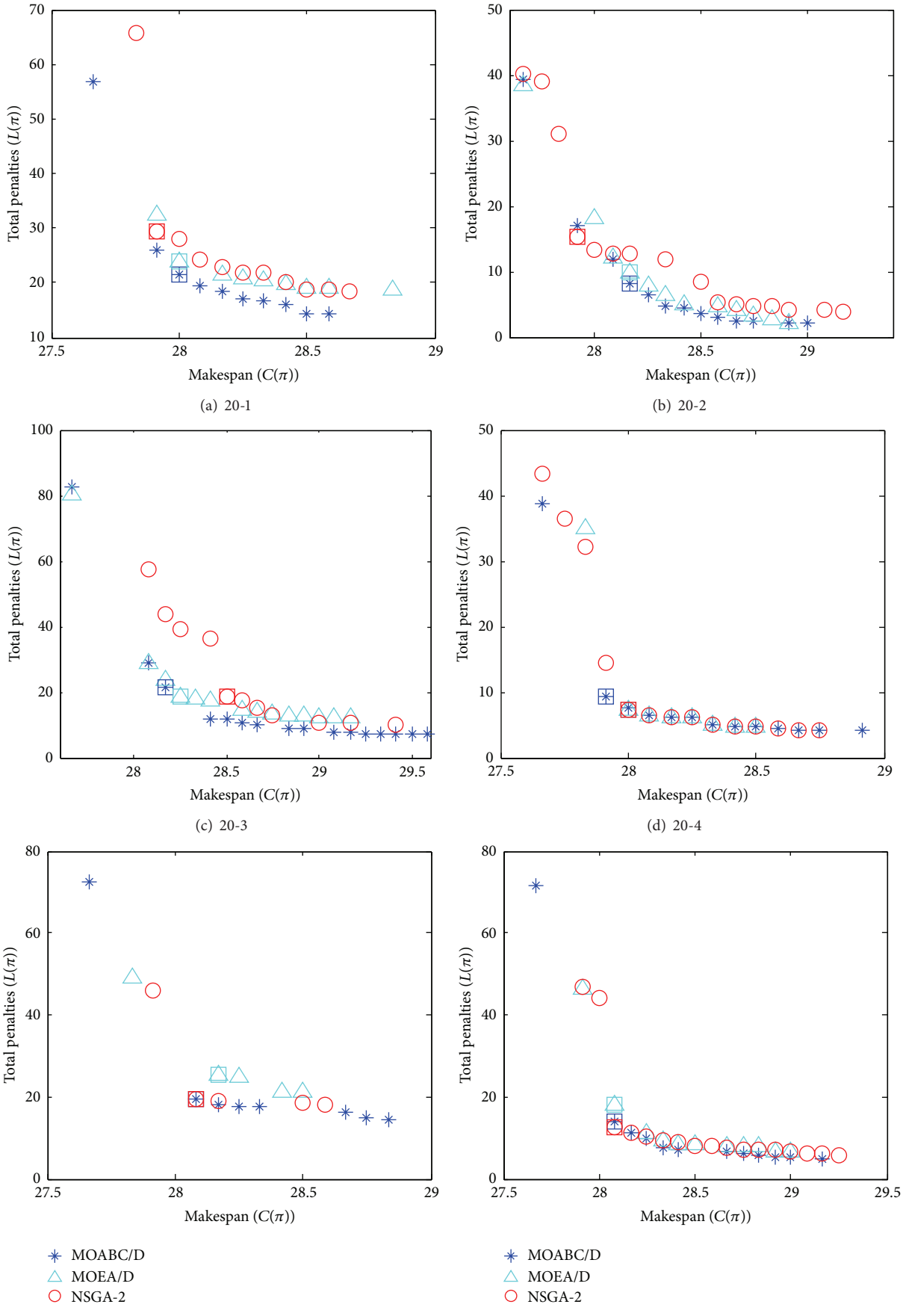
$x = C(\pi_i) - D(\pi_i)$ is the tardiness for the ith order. If the completed time is earlier than the estimated melting date, it would be a negative number, which also means earliness. The setting is based on the following consideration: if the order is completed a little earlier, there is no penalty. If it is completed much earlier than the estimated melting date, we set lesser penalty due to the inventory cost. If the order is delayed, we will set greater penalty according to the tardiness time.

We generated 6 instances for each of the three kinds of order data mentioned above and tested the MOABC/D, MOEA/D, and NSGA-II algorithms on these instances. In MOEA/D and NSGA-II, SWAP and INSERT methods are used to insure that the solutions are feasible. And neighborhood search introduced from ABC mentioned above is only employed in MOABC/D. Parameters are set as follows: in all three algorithms, population size $S = 50$. The termination criterion is evaluation count of functions reaching 50000 times. In MOABC/D and MOEA/D, amount of neighbors $N_n = 10$ and maximum update times $N_u = 2$. In NSGA-II, crossover probability $P_c = 0.9$. All algorithms are coded in MATLAB and run in MATLAB 2010b. Computer configurations are as follows: CUP: Intel Core i3-2350M, 2.30 GHz, 2.29 GHz, RAM: 1.90 GB.

The Pareto fronts obtained by the three algorithms on these instances are shown in Figure 4. Units of total penalties and makespan are both day (some solutions are repeated and overlapped in these figures, so the amount of points shown for each algorithm is less than the population size). The comparison data between three Pareto sets are shown in Table 2. As a matter of convenience, we called the Pareto sets obtained by MOABC/D, MOEA/D, and NSGA-II Pareto-set1, Pareto-set2, and Pareto-set3, respectively. In Table 2, $DomRatio_{12}$ represents the ratio between the numbers of solutions in Pareto-set1 dominated by Pareto-set2 and the total number of solutions in Pareto-set1 itself. $DomRatio_{21}$ represents the ratio between the numbers of solutions in Pareto-set2 dominated by Pareto-set1 and the total number of solutions in Pareto-set2 itself. Similarly, $DomRatio_{13}$ and $DomRatio_{31}$ represent the dominating ratio between Pareto sets obtained by MOABC/D and NSGA-II. As we are concerned mainly with the performance of MOABC/D compared with the other two algorithms, comparisons between MOEA/D and NSGA-II are ignored. $\text{Max}(\mu_1)$, $\text{Max}(\mu_2)$, and $\text{Max}(\mu_3)$ are the maximum value of μ for solutions in Pareto-set1, Pareto-set2, and Pareto-set3, which correspond to the best solutions in each Pareto set according to the fuzzy set method mentioned in Section 4.3. In the process of selecting the best, we combined the three Pareto sets to calculate the μ value. The better or best ones in each comparison in Table 2 are marked as bold. The solutions with maximum μ values are also marked using symbol "□."

As seen in Table 2, $DomRatio_{12}$ and $DomRatio_{13}$ are much smaller than $DomRatio_{21}$ and $DomRatio_{31}$ on most instances, which means that Pareto set obtained by MOABC/D outperforms that of the other two algorithms on these instances to some degree. Results shown in Figure 4 confirm the conclusion, too. $DomRatio_{12}$ is less than $DomRatio_{21}$ on 16 instances out of 18. The values of $DomRatio_{12}$ are 0 on 10 instances, which indicates that none of solutions in Pareto-set1 is dominated by Pareto-set2 on these instances. Furthermore, on instances 20-1, 30-2, 30-3, 30-5, 40-2, and 40-4, $DomRatio_{12}$ is 0 and $DomRatio_{21}$ is 1, which indicates that solutions in Pareto-set2 are all dominated by Pareto-set1 on these instance, as seen in Figures 4(a), 4(h), 4(i), 4(k), 4(n), and 4(p). The domination situations between Pareto sets of MOABC/D and NSGA-II are similar to MOABC/D and MOEA/D. $DomRatio_{13}$ is less than $DomRatio_{31}$ on all 18 instances. $DomRatio_{13}$ equals 0 on 10 instances and $DomRatio_{31}$ equals 1 on 5 instances among the 10. $\text{Max}(\mu_1)$ is the biggest among $\text{Max}(\mu_1)$, $\text{Max}(\mu_2)$, and $\text{Max}(\mu_3)$ on 16 instances among all 18 (MOABC/D and NSGA-II tied for first place on instances 20-5 and 30-3 and MOABC/D and MOEA/D tied for first place on instance 30-4), which also indicates that MOABC/D has excellent performance among the three algorithms.

Domination situation on instances with different order quantities did not show significant distinctions. However, the maximum μ values on instances with 20 orders are much smaller than on instances with order quantity of 30 and 40.

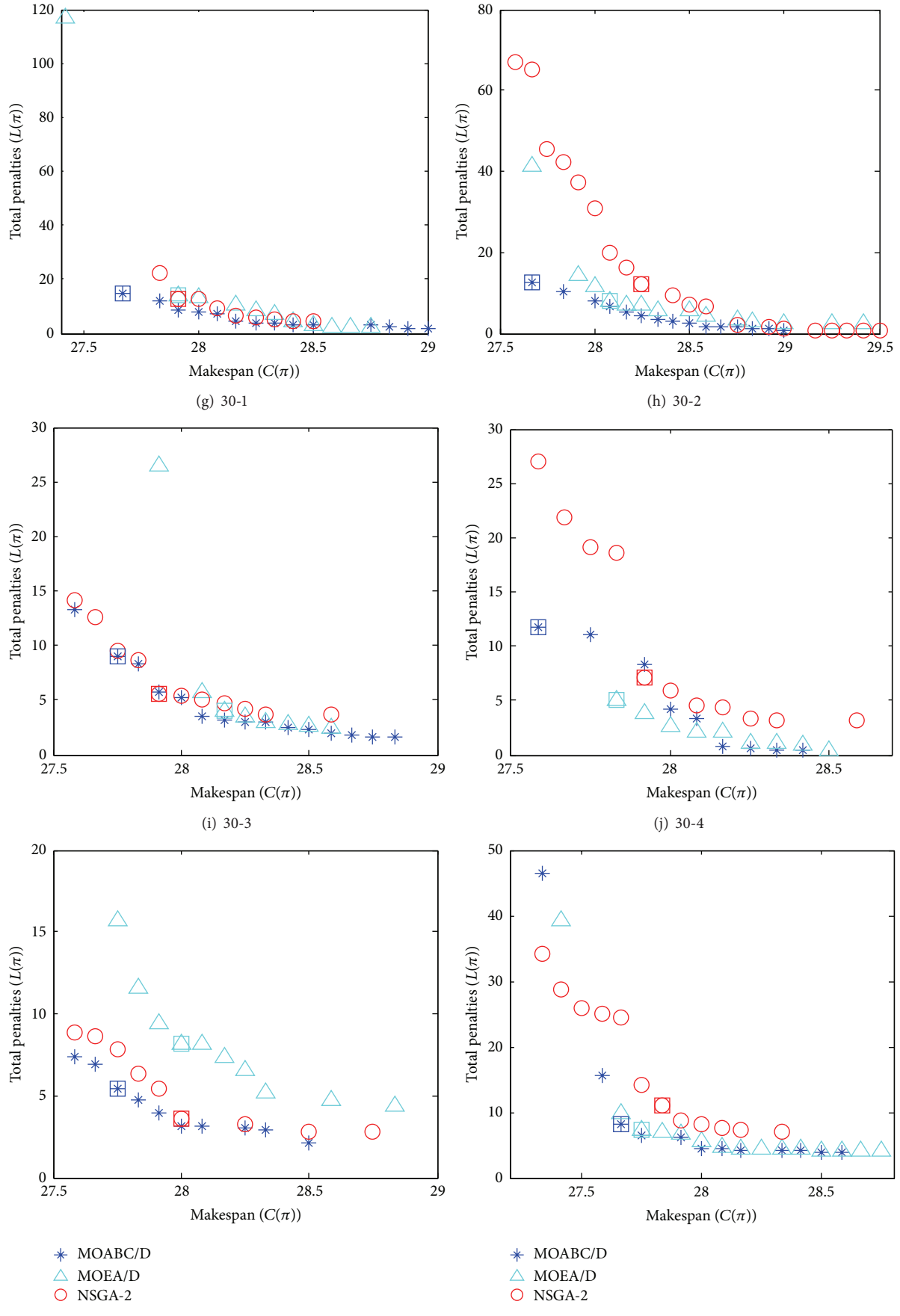


(a) 20-1 (b) 20-2

(c) 20-3 (d) 20-4

(e) 20-5 (f) 20-6

Figure 4: Continued.



(g) 30-1 (h) 30-2

(i) 30-3 (j) 30-4

(k) 30-5 (l) 30-6

Figure 4: Continued.

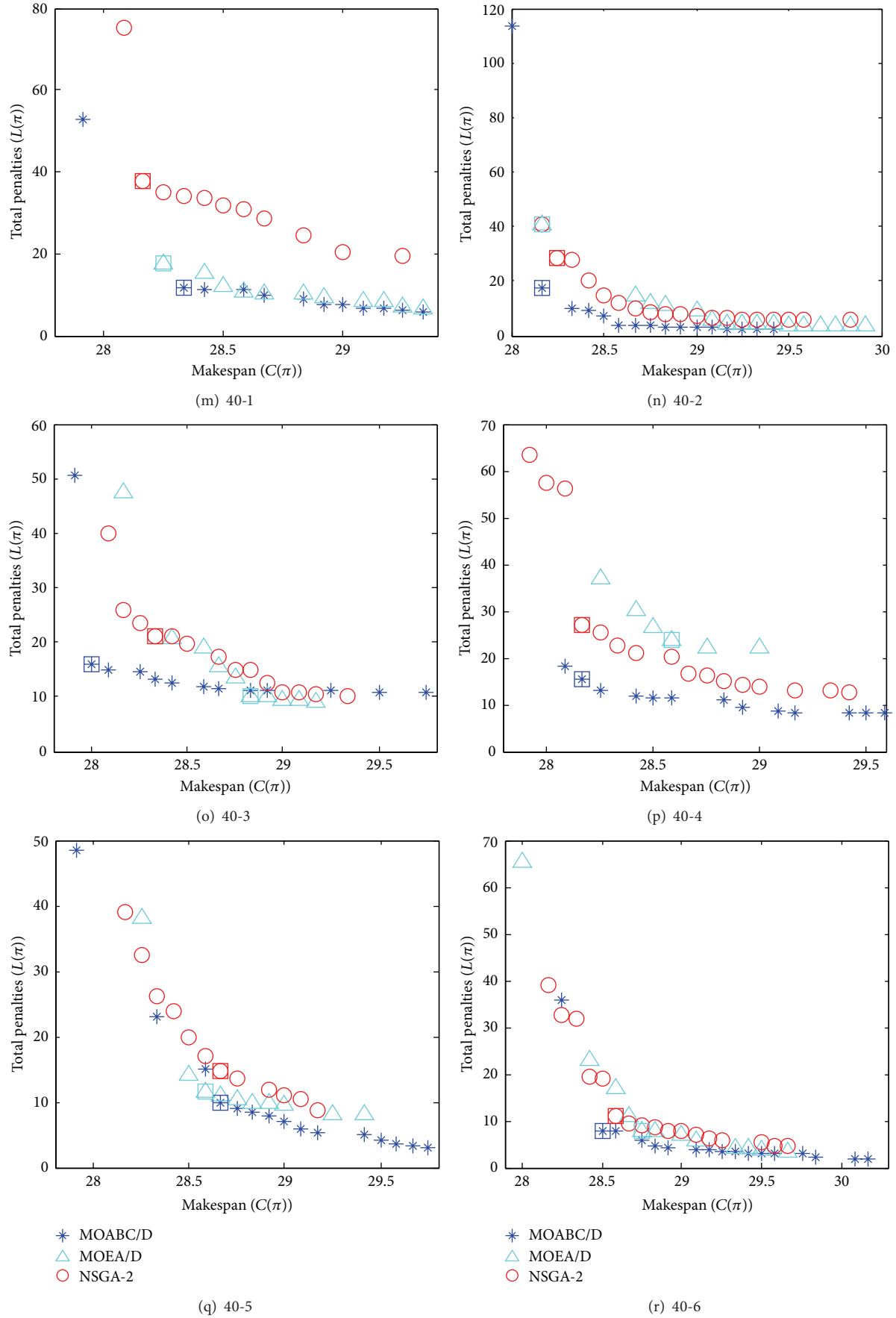


Figure 4: Pareto front obtained by MOABC/D, MOEA/D, and NSGA-II on instances (unit: day).

Table 2: Comparison of results obtained by MOABC/D, MOEA/D, and NSGA-II.

Instances	$DomRatio_{12}$	$DomRatio_{21}$	$DomRatio_{13}$	$DomRatio_{31}$	Max(μ_1)	Max(μ_2)	Max(μ_3)
20-1	**0**	1	**0**	1	**0.0095**	0.0092	0.0090
20-2	**0.0227**	0.9730	**0.0227**	0.7250	**0.0098**	0.0095	0.0097
20-3	**0.0256**	0.9118	**0**	1	**0.0105**	0.0104	0.0096
20-4	0.0588	**0**	**0.0588**	0.1500	**0.0112**	0.0111	0.0111
20-5	**0**	0.9778	**0**	0.7292	**0.0097**	0.0086	**0.0097**
20-6	**0**	0.9655	**0.0385**	0.8409	0.0124	0.0120	**0.0126**
30-1	**0.2143**	0.5833	**0**	1	**0.0162**	0.0147	0.0148
30-2	**0**	1	**0**	0.7188	**0.0151**	0.0139	0.0126
30-3	**0**	1	**0.0385**	0.8065	**0.0145**	0.0133	**0.0145**
30-4	**0.2000**	0.5714	**0.0667**	0.9091	**0.0182**	**0.0182**	0.0164
30-5	**0**	1	**0**	1	**0.0137**	0.0104	0.0132
30-6	**0**	0.9643	**0.0333**	0.7778	**0.0141**	0.0138	0.0126
40-1	**0.0455**	0.9130	**0**	0.8462	**0.0193**	0.0190	0.0162
40-2	**0**	1	**0**	1	**0.0153**	0.0135	0.0141
40-3	0.5185	**0.2381**	**0.4074**	0.7250	**0.0148**	0.0122	0.0123
40-4	**0**	1	**0**	0.8919	**0.0146**	0.0112	0.0128
40-5	**0.0435**	0.8333	**0**	0.7600	0.0175	**0.0176**	0.0162
40-6	**0**	0.9000	**0.0455**	0.7568	**0.0152**	0.0141	0.0144

This is maybe because with less order, the solution space is smaller as the dimension of solution is smaller, and more solutions are gathered near the best solutions. The values of μ are smaller accordingly. Comparison of MOABC/D and MOEA/D also proved that the neighborhood search strategy is effective in the optimization process.

It has to be mentioned that the domination situation and the priority of maximum value of μ for the Pareto sets are not completely the same. Such as on instance 20-4, several solutions obtained by the three algorithms are the same and overlapped in Figure 4(d). On this instance, there is one solution obtained by MOABC/D dominated by that of MOEA/D (on the solutions whose makespan equals 28.25, the total penalties of the solution obtained by MOABC/D is a little larger than the other algorithms). However, Max(μ_1) is bigger than Max(μ_2) and Max(μ_3), which indicates that the best solution obtained by MOABC/D (selected by fuzzy set method) is better than the other two algorithms. On instance 40-3, about half of solutions obtained by MOABC/D are dominated by that of MOEA/D. However, these solutions are concentrated on the side of minimizing the makespan. In the area of balancing the two objectives, MOABC/D performed better and Max(μ_1) is much bigger than Max(μ_2) and Max(μ_3) on this instance finally, which can be seen in Figure 4(o). On instance 20-6, MOABC/D performed better than NSGA-II on domination situation, which can be seen both from the domination situation in Table 2 and Figure 4(f). However, the best solution selected is a little worse than NSGA-II under the fuzzy set evaluation preference. The situation on instance 40-5 is similar to that on instance 20-6. MOABC/D performed better than MOEA/D on domination situation but a little worse on the best solution selected. On other 14 instances, MOABC/D performed better than the other two algorithms both on the distributions of the nondominated solutions and the priority in the optimal selection results.

On different instances, total weighted earliness and tardiness penalties are distributed from (0, 20) to (0, 120) as the constraints of these instances are different. Makespans for different instances are similar. The difference between maximum value and minimum value of makespan on most instances is nearly 1.8 days. As the production time for each order is clear, the differences of makespan for different solutions are caused by diluting remaining melt. 1.8 days mean that the difference between solutions with maximum makespan and minimum makespan is about 22 times of diluting remaining melt. For each time of diluting remaining melt, it costs about 10,000 kg copper and energy 6000 kilowatt-hours. Considering the price of commercial power in China, reducing one time of diluting remaining melt could save the direct cost by about 4000 yuan. At the meantime, the value of 10,000 kg electrolytic copper is about 400,000 yuan. Diluting remaining melt will occupy the electrolytic copper and cause metal loss in melting and cutting process, which is also a lot of cost. Choosing appropriate scheduling scheme could reduce diluting remaining melt on the premise of delivery on time to a certain degree and save much cost for enterprises.

Order data of instance 20-1 are listed in Table 3. The five solutions with largest μ values obtained by MOABC/D on instance 20-1 are also listed in Table 4.

6. Conclusions

Melting-casting is the first process in copper alloy strip industry and is also the process where energy consumption is the largest. The melting sequence and quality will influence the succedent process directly. In this paper, we studied

Table 3: Order data of instance 20-1.

Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Alloy kind	3	4	1	1	6	6	2	2	8	7	7	3	8	4	5	3	4	1	5	5
Ingots needed	6	7	9	26	7	23	29	8	26	12	24	27	18	15	9	17	12	23	28	6
Melting date	21	19	25	7	28	12	8	1	16	5	13	22	20	23	30	4	29	11	2	14

Table 4: The best five solutions with largest μ obtained by MOABC/D on instance 20-1.

π																				$L(\pi)$	$C(\pi)$	μ
8	19	16	6	4	18	7	20	10	11	9	13	1	12	5	3	15	2	17	14	28.0000	21.4667	0.0095
8	19	16	6	4	18	7	1	10	11	9	13	20	12	5	3	15	2	17	14	27.9167	26.0833	0.0094
8	19	16	7	4	18	6	20	10	11	9	13	1	12	5	3	15	2	14	17	28.0833	19.5417	0.0093
8	19	16	6	4	18	7	20	10	11	2	9	13	1	12	5	3	15	14	17	28.1667	18.3000	0.0090
8	19	16	7	4	18	6	20	10	11	2	9	13	1	12	5	3	15	14	17	28.2500	16.8750	0.0088

the scheduling problem of melting-casting and built the multiobjective model minimizing the makespan and total weighted earliness and tardiness penalties. A new algorithm which is based on the structure of MOEA/D and the neighborhood searching strategy of ABC is proposed to solve the problem, which we named MOABC/D. In MOABC/D, encoding method and new individual producing rules are also redesigned to adapt to the scheduling model. Besides, in order to provide conveniences for the schedulers to make decision, we used fuzzy set method to sort the nondominated solutions finally according to their actual preference. Results show that MOABC/D outperforms MOEA/D and NSGA-II on most instances both on the distributions of the Pareto front and the priority in the optimal selection results. The economic benefit analysis also shows that it could help the enterprise save much cost by choosing appropriate scheduling scheme.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (nos. 71201026, 71271140, and 71471158), the Development Program for Excellent Young Teachers in Higher Education Institutions of Guangdong Province (no. Yq2013156), the Project of Department of Education of Guangdong Province (no. 2013KJCX0179), and the High Level Talent Project in Guangdong Province. And the authors are very grateful to the anonymous reviewers for their valuable suggestions and comments to improve the quality of this paper.

References

[1] W. Zhong, K. Ma, and W. Wu, *Practical Guide of Copper Processing Technology*, Metallurgical Industry Press, Beijing, China, 2007, (Chinese).

[2] X.-H. Yan, Y.-L. Zhu, and C.-X. Lu, "Order-oriented copper strip grouping and optimization," *Computer Integrated Manufacturing Systems*, vol. 17, no. 9, pp. 1938–1943, 2011 (Chinese).

[3] V. R. Gandhewar, S. V. Bansod, and A. B. Borade, "Induction furnace—a review," *The International Journal of Engineering and Technology*, vol. 3, no. 4, pp. 277–284, 2011.

[4] M. H. Fazel Zarandi and R. Gamasaee, "Type-2 fuzzy hybrid expert system for prediction of tardiness in scheduling of steel continuous casting process," *Soft Computing*, vol. 16, no. 8, pp. 1287–1302, 2012.

[5] J. Zhao, Q. Liu, W. Wang, Z. Wei, and P. Shi, "A parallel immune algorithm for traveling salesman problem and its application on cold rolling scheduling," *Information Sciences*, vol. 181, no. 7, pp. 1212–1223, 2011.

[6] T. Zhang, Y.-J. Zhang, Q. P. Zheng, and P. M. Pardalos, "A hybrid particle swarm optimization and tabu search algorithm for order planning problems of steel factories based on the make-to-stock and make-to-order management architecture," *Journal of Industrial and Management Optimization*, vol. 7, no. 1, pp. 31–51, 2011.

[7] A. Atighehchian, M. Bijari, and H. Tarkesh, "A novel hybrid algorithm for scheduling steel-making continuous casting production," *Computers & Operations Research*, vol. 36, no. 8, pp. 2450–2461, 2009.

[8] N. Geismar, "Single machine scheduling," in *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley & Sons, 2011.

[9] C. A. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[10] W. Zou, Y. Zhu, H. Chen, and B. Zhang, "Solving multiobjective optimization problems using artificial bee colony algorithm," *Discrete Dynamics in Nature and Society*, vol. 2011, Article ID 569784, 37 pages, 2011.

[11] A. R. Yildiz and K. N. Solanki, "Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach," *International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1–4, pp. 367–376, 2012.

[12] S. M. Yang, D. G. Shao, and Y. J. Luo, "A novel evolution strategy for multiobjective optimization problem," *Applied Mathematics and Computation*, vol. 170, no. 2, pp. 850–873, 2005.

[13] T. W. Athan and P. Y. Papalambros, "A note on weighted criteria methods for compromise solutions in multi-objective optimization," *Engineering Optimization*, vol. 27, no. 2, pp. 155–176, 1996.

[14] L. G. Vargas, "An overview of the analytic hierarchy process and its applications," *European Journal of Operational Research*, vol. 48, no. 1, pp. 2–8, 1990.

[15] C. M. Fonseca and P. J. Fleming, "Genetic algorithm for multi objective optimization: formulation, discussion and generation," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest, Ed., pp. 416–423, Morgan Kauffman Publishers, Urbana-Champaign, Ill, USA, June 1993.

[16] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[17] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto-envelope based selection algorithm for multi-objective optimization," in *Parallel Problem Solving from Nature, PPSN VI*, M. Schoenauer, K. Deb, G. Rudolph et al., Eds., Lecture Notes in Computer Science, pp. 869–878, Springer, Berlin, Germany, 2000.

[18] C. A. C. C. Coello and G. T. Pulido, "A micro-genetic algorithm for multiobjective optimization," in *Evolutionary Multi-Criterion Optimization: First International Conference, EMO 2001 Zurich, Switzerland, March 7–9, 2001 Proceedings*, vol. 1993 of *Lecture Notes in Computer Science*, pp. 126–140, Springer, Berlin, Germany, 2001.

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

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

[21] G. Moslehi and M. Mahnam, "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search," *International Journal of Production Economics*, vol. 129, no. 1, pp. 14–22, 2011.

[22] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[23] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[24] L. Ma, K. Hu, Y. Zhu, and H. Chen, "A hybrid artificial bee colony optimizer by combining with life-cycle, Powell's search and crossover," *Applied Mathematics and Computation*, vol. 252, pp. 133–154, 2015.

[25] K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic, Norwell, Mass, USA, 1999.

[26] J. Wang, Q. Duan, Y. Jiang, and X. Zhu, "A new algorithm for grid independent task schedule: genetic simulated annealing," in *Proceedings of the World Automation Congress (WAC '10)*, pp. 165–171, September 2010.

[27] B. Liu, L. Wang, and Y.-H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 18–27, 2007.

[28] M. Farina and P. Amato, "A fuzzy definition of 'optimality' for many-criteria optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 34, no. 3, pp. 315–326, 2004.

[29] H. Zhang, Y. Zhu, W. Zou, and X. Yan, "A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production," *Applied Mathematical Modelling*, vol. 36, no. 6, pp. 2578–2591, 2012.