Coil Batching to Improve Productivity and Energy Utilization in Steel Production

Lixin Tang [*] and Ying Meng

Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang, 110004, China

Zhi-Long Chen

Robert H. Smith School of Business, University of Maryland, College Park, Maryland 20742, USA

Jiyin Liu

School of Business and Economics, Loughborough University, Leicestershire, LE11 3TU, UK

Abstract: This paper investigates a practical batching decision problem that arises in the batch annealing operations in the cold rolling stage of steel production faced by most large iron and steel companies in the world. The problem is to select steel coils from a set of waiting coils to form batches to be annealed in available batch annealing furnaces and choose a median coil for each furnace. The objective is to maximize the total reward of the selected coils less the total coil-coil and coil-furnace mismatching cost. For a special case of the problem that arises frequently in practical settings where the coils are all similar and there is only one type of furnace available, we develop a polynomial-time dynamic programming algorithm to obtain an optimal solution. For the general case of the problem which is strongly NP-hard, an exact branch-and-price-and-cut solution algorithm is developed using a column and row generation framework. A variable reduction strategy is also proposed to accelerate the algorithm. The algorithm is capable of solving medium size instances to optimality within a reasonable computation time. In addition, a tabu search heuristic is proposed for solving larger instances. Three simple search neighborhoods as well as a sophisticated variable depth neighborhood are developed. This heuristic can generate near-optimal solutions for large instances within a short computation time.

Using both randomly generated and real-world production data sets, it is shown that our algorithms are superior to a typical rule-based planning approach used by many steel plants. A decision support system that embeds our algorithms was developed and implemented at Baosteel to replace their rule-based planning method. The use of the system brings significant benefits to Baosteel, including an annual net profit increase of at least 1.76 million US Dollars, and a large reduction of standard coal consumption and carbon dioxide emissions.

Key words: steel production, batch annealing, batching decisions, integer programming, dynamic programming, branch-and-price-and-cut, tabu search

[*] Corresponding author. Email: qhjytlx@mail.neu.edu.cn

1. Introduction

The steel industry has been one of the pillar industries in the world economy and is a powerful symbol of industrialization, urbanization and economic development, leading the development of other industries such as construction, shipbuilding, machinery, home appliances, and automotive. However, globally, the steel industry is also one of the largest industrial energy consumers and one of the largest contributors to greenhouse gas and air pollution. Consequently, for most steel companies, improving energy utilization and reducing environmental impact become equally important as improving productivity and profitability.

Iron and steel production is a complicated multistage process that mainly consists of iron-making, steel-making, hot rolling and cold rolling stages, where batch processing is the most commonly used production mode. A detailed description of various production processes in integrated steel production can be found in Tang *et al*. (2001). Cold rolled products are of the highest value-added among all steel products. Therefore, cold rolled products are a key to increasing profitability. The global annual output of cold rolled products accounts for about 40 percent of the total steel product output. The production process of cold rolled products consists of pickling, rolling, batch annealing, temper rolling, and some post-treatment steps. Batch annealing is a key operation which can improve the mechanical properties of cold rolled products. During the annealing operation, steel coils are first heated and soaked in a high temperature (i.e. 600-1200°C), and then cooled to room temperature. This can change the structure of crystal grain of steel coils and hence improve their thermodynamic stability and guarantee their performance. However, annealing is a main energy-consuming operation in the cold rolling stage; it consumes a huge amount of energy and resources, such as water, electricity, coal gas and protective gas. Thus, the batch annealing process is one of the bottlenecks at most steel plants, and improving its efficiency becomes crucial to increase a steel company's profitability and energy utilization.

Steel coils waiting to be annealed in annealing furnaces generally have different sizes and physical characteristics according to specific requirements of customer orders. The structure of a coil and that of an annealing furnace are shown in Figures 1 and 2, respectively. Each furnace consists of a base for holding the coils to be annealed and two layers of covers – inner cover and outer cover. Coils to be annealed in a furnace are stacked one on top of another, with a convector plate underneath each coil to help the heat reach inside the coils. The height of a convector plate in a steel plant for any type of furnace is always the same (70 mm). There are four types of protective gas atmospheres used within a furnace: nitrogen hydrogen (NH), pure hydrogen (HH), pure nitrogen (NN) and pure argon (AR). By both gas atmosphere and the inner-diameter of the inner cover, batch annealing furnaces can be classified into several different types, e.g. NH-small, NH-medium, NH-large, HH-small, HH-medium, HH-large, and so on.

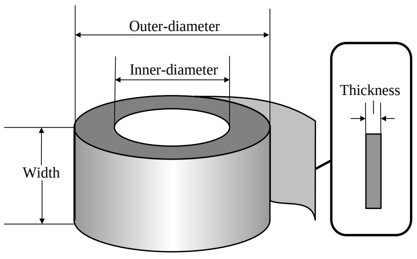


Figure 1. The structure of a coil

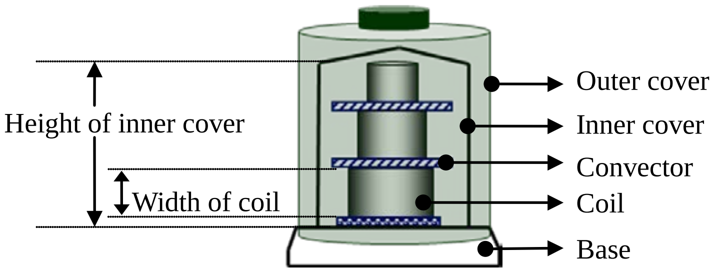


Figure 2. The structure of a batch annealing furnace

The batch annealing process can be described as follows. After a batch of coils is loaded onto a furnace and protective gas atmosphere is filled in to prevent oxidation, a series of heating and cooling operations are executed following a *temperature control curve* of the furnace. For a loaded furnace, the temperature control curve represents the temperature of the gas atmosphere in the furnace over time. Although each coil has its own *annealing curve* which the annealing process should follow for annealing this coil in order to guarantee its highest quality, the temperature of a furnace can only be set following one control curve for a batch of coils. Ideally, each loaded furnace should use a temperature control curve that can maximize the overall quality of the coils in the furnace at minimum energy consumption. However, it would require a series of technological experiments and tests involving a lot of time and cost in order to identify such an optimal temperature control curve. Therefore, it is a common practice that one coil is selected from the batch as the *median coil* whose annealing curve is used to set the temperature control curve of the furnace. The total time needed for annealing a batch of coils varies from 48 to 76 hours, depending on the characteristics of the median coil in the batch.

In most steel plants, planners make coil batching decisions on a shift-by-shift basis such that the decisions in each 8-hour shift are made at the end of the previous shift by considering all furnaces that become available in the planning shift and all coils that are waiting to be annealed at the time of decision. In general, the number of coils waiting to be annealed prior to the start of a shift can vary from tens to hundreds, depending on available production capacity, market demand, and the schedule of batch annealing operations. Market demand for batch annealed coils can vary significantly from month to month. During low demand months when the number of coils to be annealed in each shift is low, some available furnaces may be shut down intentionally to save energy consumption and to maximize average charging weight of a furnace. Since batch annealing takes a long time to complete, the furnaces that begin annealing a batch of coils in a shift are not available until several shifts later. The number of available furnaces in a shift is usually about one tenth of the number of coils to be annealed. In most cases, the available furnaces are not enough to cover all the coils currently waiting to be annealed such that some coils may have to wait for a future shift to be annealed.

The batching decision problem can be generally described as follows. A set of coils with different physical characteristics are to be annealed using a number of available empty furnaces. There are two sets of decisions to make. The first is to select a subset of coils to form batches, each to be loaded into an available empty furnace. The second is to choose one of the coils in each furnace as a median coil based on which a temperature control curve is set for the furnace. In making these decisions, planners are faced with many physical constraints and business rules (described in Section 3.1). Planners usually consider two objectives. Their first objective is maximizing the total reward of the coils covered, where the reward of a coil is determined by several factors including its weight, due date, and its relationship with other coils. The total reward of the coils annealed in a furnace is an aggregate performance measure that takes into account the charging weight and customer service, where the *charging weight* of a furnace is defined as the total weight of coils annealed in the furnace. In practice, the total energy consumption for annealing a batch of coils in a furnace is independent of the total charging weight of the furnace. By maximizing the total reward, the charging weight and energy utilization of a furnace are also maximized in most cases. Their second objective is minimizing the total cost due to the waste of energy and impact on coil quality caused by mismatching of coils in the same batch and mismatching between furnaces and the coils inside. In practice, the differences of physical characteristics between the median coil and the other coils are used to measure the level of mismatching between coils in the same batch. Both rewards and mismatching costs are measured in monetary terms at Baosteel and hence can be considered together.

As discussed earlier, following the practice by most steel plants including Baosteel, the batching decision problem defined above is for one shift only. Theoretically, a better solution could be obtained if multiple shifts are considered jointly. However, if multiple shifts are considered together, one would need to forecast future coil arrivals prior to each of the future shifts involved, including the specific types of coils to arrive, the quantity of each type and the time they will arrive. The benefit from a multi-shift model will depend on the accuracy of the forecast of future coil arrivals, especially due to the combinatorial nature of the problem. Unfortunately, in a complex context like Baosteel, there will be forecast errors due to the natural variability in the upstream production processes prior to batch annealing. Hence, we choose to use the single-shift model as defined earlier, as it is compatible with Baosteel’s current practice, and, as shown later in the paper, using such a model already provides substantial opportunity for improvement. We believe that in some practical settings, e.g., when there are many urgent coils waiting to be annealed, using the single-shift model can already generate near-optimal solutions for the multi-shift problem. In the future, if planners can gather accurate information about coils to arrive in subsequent shifts, the problem could be extended to include multiple shifts to generate potentially better batching decisions.

In most steel plants, coil batching decisions are either made manually by planners based on their experience and intuition or by computerized decision support systems that mainly rely on greedy rules that consider coils batch-by-batch instead of jointly (described in Section 7.1). Although planners may have extensive experience, it is very hard to construct effective batching plans manually due to the complex constraints involved. Similarly, it can be expected that greedy rules generally do not work well for such a complex problem which, as will be seen later, is a strongly NP-hard combinatorial optimization problem.

In this paper, we formulate the batching decision problem as an optimization problem with a single objective function which is the total reward of the selected coils minus the total mismatching cost. We first propose a polynomial-time dynamic programming algorithm to solve a special case of the problem that arises frequently in practical settings. For the general case of the problem, we develop a branch-and-price-and-cut algorithm to obtain optimal solutions for medium size problems and a tabu search heuristic to obtain near-optimal solutions for large scale problems. We have developed a batching decision support system for Baosteel that contains all of our algorithms. The use of this system, as benchmarked against their previous solution approach, has brought significant benefits to Baosteel.

The rest of this paper is structured as follows. Section 2 reviews related literature. Section 3 defines the batching decision problem formly and formulates it as a binary integer program. Section 4 describes the dynamic programming algorithm for the special case of the problem. Sections 5 and 6 describe the branch-and-price-and-cut exact solution algorithm and the tabu search heuristic for the general problem, respectively. Section 7 reports the computational results of the developed algorithms. Section 8 estimates the benefits brought to Baosteel by this research. Finally we conclude the paper in Section 9.

2. Related Literature and Problem Complexity

Batching problems arising in other industries that we are aware of, including semiconductor and chemical industries (e.g., Lee *et al.* 1992, Prasad and Maravelias 2008), are very different from the batching problem we study in this paper because of different technological and management constraints involved in different industries. Therefore, our literature review is focused on the steel industry. Optimization tools have been applied to a large variety of decision problems in the steel industry (Dutta and Fourer 2001). Tactical and operational planning problems are a subset of such problems. These problems can generally be classified as scheduling or batching problems. Scheduling is concerned with assignment of jobs to machines and sequencing and timing of jobs. Batching is concerned with grouping jobs into batches for joint processing.

To the best of our knowledge, all the batching related problems in the iron and steel industry reported in the literature arise from operations other than batch annealing, and hence have a different structure from the problem we study in this paper. Kalagnanam *et al.* (2000) study a surplus inventory matching problem arising in operations planning in the process industry. In the case of a steel plant, this problem is to assign surplus slabs to orders with the objective of maximizing the total weight of surplus slabs used. Balakrishnan and

Geunes (2003) study a flexible demand assignment problem that is to decide which slabs to be selected and processed into plates, which plates to be assigned to each selected slab, and what finished weight to produce in the stainless steel plate production process. Chu and Antonio (1999) address a real-life uni-dimensional cutting stock problem where given input rods need to be cut into output rods ordered by customers. The above two assignment problems and cutting stock problem can be viewed as batching problems because they involve grouping the slabs (or plates, output rods) into batches and assigning them to orders (or slabs, input rods). In these problems, there is no compatibility issue between the slabs, plates or output rods (i.e., any slabs, plates, or output rods can be assigned to the same batch), and there is no such concept as median coils in our problem.

Tang and Jiang (2009) investigate a charge batching problem arising from the steel making stage. The problem is to batch primary order requirements into various production charges subject to some processing constraints and composite batch conditions. Naphade *et al*. (2001) study a batching and scheduling problem for the melting process in steelmaking which is to select ingots to be processed and decide how to batch the selected ingots for melting in a given horizon. Tang and Wang (2008) consider an order batching problem in the steelmaking process and a charge batching problem in the continuous casting process. The order (charge) batching problem is to decide how to consolidate orders into charges (casts) considering the compatibility of any two orders (charges) in the same batch. Tang *et al.* (2014) investigate a joint charge batching and casting width selection problem arising in the continuous casting operation at Baosteel. Tang *et al*. (2011) study a batching problem in the melting process to group orders into batches and specify the size and the grade of slabs for each batched order. In all the above reviewed batching problems, all batches are processed by identical equipment and hence there is no compatibility issue between the items and the equipment. Furthermore, in these problems there is also no such concept as median coils in our problem.

Many studies on scheduling problems that arise in steel production have also been reported in the literature. Since scheduling is not the focus of this paper, we only briefly review some representative papers below. Vonderembse and Haessler (1982) study a ripper scheduling problem resulting from the casting stage of steel production. Tang *et al*. (2000) study a hot rolling scheduling problem. Möhring *et al*. (2003) investigate a resource-constrained project scheduling problem which has a variety of applications in the steel industry. König *et al*. (2007) study a stacking problem related to storage planning of steel slabs in steel production. Höhn *et al*. (2012) study a no-wait flowshop scheduling problem that arises from the continuous casting operations in steel production. Höhn *et al*. (2011) consider a scheduling problem for color coating operations in the cold rolling stage. Moon and Hrymak (1999) investigate a short-term scheduling problem for the batch annealing process in the cold rolling stage of steel production. Tang *et al*. (2009) study a scheduling problem in batch annealing operations in the cold rolling stage which is to assign inner covers and outer covers to given batches and schedule a single crane for minimizing the completion time of the last batch of coils. Although the problems studied by Moon and Hrymak (1999) and Tang *et al*. (2009) are both related to batch annealing operations, the batching of steel coils is a known input in their scheduling problems.

As described earlier, in our problem, there are two sets of decisions: assigning coils to furnaces and selecting a median coil for each furnace, and there are two parts in the objective: the total reward of coils assigned to a given number of furnaces and the total cost of mismatching between the coils and the furnaces and between the median coil and the other coils in each furnace. With the only consideration of the first part of the objective and one set of the decisions (coil assignment), our problem can be viewed as the multiple knapsack problem if we view the different types of furnaces in our problem as different types of knapsacks in the latter problem. The multiple knapsack problem is known to be strongly NP-hard (Martello and Toth 1990). Hence our problem is also strongly NP-hard. There are several existing studies on the multiple knapsack problem. Chekuri and Khanna (2006) prove that the multiple knapsack problem does not admit an FPTAS even for the case with two knapsacks and give a polynomial time approximation scheme for the multiple knapsack problem. Dawande *et al*. (2000) investigate the multiple knapsack problem with assignment restrictions where the objective is to maximize assigned weight and minimize utilized capacity.

With the only consideration of the first part of the objective and one set of decisions (selection of median coils), our problem can be viewed as the p-median problem with capacity constraints if we view the median coils in our problem as the medians in the latter problem. The p-median problem is known to be NP-hard (Megiddo and Supowit 1984). In fact, it is shown (Lin and Vitter 1992) that there is no polynomial-time algorithm with a constant worst-case bound for the p-median problem. Since our problem is more general, this result also applies to our problem.

Therefore, our problem can be seen as a combination of the multiple knapsack problem and the p-median problem, and inherits the complexity of both problems. Different from the multiple knapsack problem, the decision of assigning a coil to a furnace is not only decided by the reward of the coil, but also by the selection of the median coil in that furnace. Different from the p-median problem, both clustering and assignment of clusters to empty furnaces are considered jointly in our problem. Because of these differences, no existing solution algorithms for the multiple knapsack problem or p-median problem can be applied to our problem.

3. Problem Description and Formulation

3.1. Constraints and Requirements

To ensure that each coil in a batch receives the level of annealing required, coils in the same batch must have the same or similar characteristics with the median coil and the annealing of the batch must be done in an appropriate type of furnace. These considerations can be described as the following constraints and requirements, which to our knowledge are adopted by many steel plants.

3.1.1 Equipment Constraints

The outer-diameter of each coil in the batch must be smaller than the inner-diameter of the inner cover of the furnace where the batch is annealed. The total height of a batch, which is equal to the sum of the total width of all the coils and the total height of all the convector plates in the batch, must not exceed the height of the inner cover of the furnace. In practice, the designed weight capacity of a furnace is always large enough to hold any number of coils as long as the height capacity is not violated. Therefore, there is no explicit weight capacity constraint in the problem.

3.1.2 Matching Constraints and Requirements

To maximize the quality of annealed coils, only compatible coils can be annealed in the same furnace. Whether two coils are compatible or not is judged by the relevant characteristics of the coils including annealing curve, thickness, outer-diameter, surface flatness and steel grade. For each coil in a batch, if the relevant characteristics of the coil are not identical to but compatible with those of the median coil in the batch, a cost is incurred due to the mismatching between the characteristics of these two coils.

Besides the above described matching requirements between coils, there are also matching requirements between coils and the protective gas atmosphere in annealing furnaces. According to its annealing curve index, a coil may have to be processed in a furnace with a particular type of protective gas or with one of several types of protective gas but with some types being more appropriate than others. Mismatching between a coil and a furnace is discouraged through a penalty which is assigned based on the annealing curve index of the coil and the protective gas atmosphere of the furnace.

3.1.3 Management Considerations

In addition to the technical requirements described above, planners also need to consider management issues concerning customers and internal logistics, including due dates, relationship between coils, and contract accumulation times and storage times of coils. From these factors, we can judge if a coil should be processed as quickly as possible. We quantify these management issues and represent them collectively using a priority index (PRI). Coils with a larger value of PRI are given a higher priority for batching. In Appendix A, we describe how the PRI values and the values of some other parameters are set at Baosteel.

3.2. Problem Formulation

In this subsection, we define mathematical notation and formulate the batching decision problem as a 0-1 integer programming model.

Parameters:

N	Set of n coils waiting to be annealed, $N = \{1, 2, \ldots, n\}$;
P	Set of all available annealing furnaces;
O_j	Set of coils which can be loaded into furnace j following the equipment and matching constraints described in Section 3.1 ($\bigcup_{j\in P} O_j = N$);
R_k	Set of coils which can be loaded together with coil k following the matching constraints described in Section 3.1;
Q_i	Set of available annealing furnaces where coil i can be loaded, $Q_i = \{j \in P \mid i \in O_j\}$;
H	Height of the inner cover of a furnace;
g_i	Weight of coil i;
h_i	Sum of the width of coil i and the height of a convector plate;
f_i	PRI value of coil i, which is determined by its due date, relationship with other coils, contract accumulation times and storage times, as described in Section 3.1.3;
F_i	Reward for covering coil i in the batching plan, which is the weighted sum of the PRI value and the weight of the coil, i.e., $\rho f_i+(1-\rho)g_i$, where $0 < \rho < 1$ is the weighting of PRI factor;
C_{1ij}	Mismatching cost between coil i and furnace j, representing the mismatching between the annealing curve index of coil i and the type of protective gas atmosphere in furnace j, as described in Section 3.1.2;
C_{2ik}	Mismatching cost between coil i and coil k, representing the mismatching of these coils in terms of annealing curve index, thickness, outer-diameter, surface flatness, and steel grade, as described in Section 3.1.2.

Decision Variables:

X_{ij} = 1 if coil $i\in N$ is loaded into annealing furnace $j\in Q_i$, and 0 otherwise

Y_{ikj} = 1 if coil $i\in N$ is loaded into annealing furnace $j\in Q_i$ whose median is coil $k\in R_i$, and 0 otherwise

The batching decision problem can then be formulated as an integer programming model below.

(BDP)

$$\text{Max } Z = \sum_{j\in P}\sum_{i\in O_j} F_i X_{ij} - \sum_{j\in P}\sum_{i\in O_j}(C_{1ij}X_{ij} + \sum_{k\in O_j\cap R_i} C_{2ik}Y_{ikj}) \tag{1}$$

Subject to

$$\sum_{j\in Q_i} X_{ij} \le 1 \qquad \text{for } i\in N \tag{2}$$

$$\sum_{i\in O_j} h_i X_{ij} \le H \qquad \text{for } j\in P \tag{3}$$

$$\sum_{k\in O_j} Y_{kkj} = 1 \qquad \text{for } j\in P \tag{4}$$

$$Y_{ikj} \le Y_{kkj} \qquad \text{for } j\in P,\ k\in O_j,\ i\in O_j \cap R_k \tag{5}$$

$$X_{ij} = \sum_{k\in O_j\cap R_i} Y_{ikj} \qquad \text{for } j\in P,\ i\in O_j \tag{6}$$

$$X_{ij} \in \{0,1\} \qquad \text{for } j \in P,\ i \in O_j \tag{7}$$

$$Y_{ikj} \in \{0,1\} \qquad \text{for } j \in P,\ k \in O_j,\ i \in O_j \cap R_k \tag{8}$$

The objective function (1) is the total reward minus the total mismatching cost. Constraint (2) ensures that each coil can be loaded into at most one furnace. Constraint (3) is the height capacity constraint for each furnace. Constraint (4) requires each furnace to have exactly one median coil. Constraints (5) and (6) mean that if a coil is loaded into a furnace, it must be in the same batch as the median coil of the furnace. Constraints (7) and (8) set the binary integer requirements for the variables.

For a large problem instance with 200 coils and 20 furnaces, this formulation contains more than 100,000 binary integer variables and 40,000 constraints. A commercial IP solver such as CPLEX cannot deal with such a large scale problem due to memory overflow. For a medium problem instance with 100 coils and 10 furnaces, as will be seen later when we report computational results, a commercial IP solver cannot obtain an optimal solution in two hours. However, in practice, there is a time window of less than one hour at the end of a shift within which the batching decisions for the next shift must be made. This motivates us to design specialized solution methods to solve the problem more efficiently.

For ease of presentation in the sections that follow, we define the following concept: a feasible *batching scenario* for a furnace is a subset of coils that can be loaded into the furnace satisfying all the constraints discussed earlier, with one of the coils specified as the median coil. Mathematically, a feasible batching scenario for furnace j is the part of solution to the formulation BDP that specifies the values of all the variables related to furnace j, i.e., X_{ij} and Y_{ikj} variables, for all i and k.

4. A Frequently Occurring Special Case

Some small to medium steel plants (e.g. China's Tiantie and Benxi Steel Companies) only produce annealed coils with a similar dimension (i.e., width, thickness, and steel grade) using only one type of furnace. Consequently, considering the height capacity of annealing furnaces, the maximum number of coils which can be assigned to each furnace is fixed, i.e. $\lfloor H/h_i \rfloor = \lfloor H/h_k \rfloor$ for any two coils i and k to be annealed. Let r be the maximum number of coils which can be assigned to each furnace.

Since all the coils will have to be annealed in a single type of furnace, the mismatching cost between a coil and any furnace is fixed. Consequently, we can ignore all mismatching costs between the coils and the furnaces. In practice, coils with similar steel grades always have similar required annealing processes, hence any coil can be batched into the same furnace. Given the small differences in width, thickness, and steel grade, mismatching between coils is mostly represented by mismatching associated with the required heat consumption during the annealing process. Let T_i denote the amount of heat consumption by coil i during the annealing process. Then the mismatching cost between two coils i and k can be calculated as $C_{2ik}=\theta|T_i-T_k|$, where θ is a constant.

For a small to medium steel plant, it is often the case that its available annealing capacity in a shift is not enough to cover all the coils from a single customer order. For ease of order management, the plant often dedicates multiple consecutive shifts to the coils from each order. Consequently, the plant is only concerned with the coils from a single order in some shifts. Therefore, we can assume that all the coils to be considered in a shift have the same due date, contract accumulation time and storage time. Thus, by the definition of priority value of a coil f_i (given in Section 3.2), the coils from the same order usually have the same priority value. Furthermore, coils with a higher weight consume more heat, and vice versa, i.e., given two coils i and k, $T_i \geq T_k$ if and only if $g_i \geq g_k$. Recall that the reward of a coil i is defined as $F_i = \rho f_i+(1-\rho)g_i$. Therefore, it is practical to assume that the reward values F_i and F_k of any two coils i and k to be considered in a shift are agreeable with their required heat consumption, i.e., if $T_i \geq T_k$, then $F_i \geq F_k$.

Based on the above discussions, we consider a special case of the batching decision problem as follows. Given n coils satisfying the agreeable condition: if $T_i \geq T_k$, then $F_i \geq F_k$ for any two coils i and k, select a subset of them to be loaded to p ($p=|P|$) given identical furnaces with a capacity of r coils each, so as to maximize the total reward minus the total mismatching cost between the selected coils.

We have the following optimality property for this special case of the problem.

LEMMA 1: For any given batching scenario for a furnace in an optimal solution, reindex the coils in the furnace as [1], [2], …, [q], such that $T_{[1]} \geq T_{[2]} \geq \ldots \geq T_{[q]}$, where $q \leq r$ is the number of coils in the furnace. The median coil for this furnace must be coil [l], where $l = \lceil q/2 \rceil$.

PROOF: See Appendix B.

In the rest of this section, all the coils are re-indexed such that $T_1 \geq T_2 \geq \ldots \geq T_n$, and $F_1 \geq F_2 \geq \ldots \geq F_n$. In any given batching scenario with q coils, if we sequence the coils in the increasing order of their indices, then by Lemma 1, the median coil is the l-th coil in the sequence, where $l = \lceil q/2 \rceil$.

For ease of presentation, in a given batching scenario, we refer to coils with indices smaller (larger) than that of the median coil as *left coils* (*right coils*). Given two sets of coils, we say that the index ranges of the two sets are *non-overlapping* if the indices of the coils in one set are all greater or all smaller than the index of any coil in the other set.

THEOREM 1: There exists an optimal solution where the index range of all the coils assigned to any furnace is non-overlapping with that of all the coils assigned to any other furnace.

PROOF: See Appendix B.

By Theorem 1, we can consider the coils in increasing order of their indices and try to add them one by one to furnaces such that if a coil is considered but not added to the current furnace, then this coil is skipped and will never be considered again. Based on this idea, we propose a dynamic programming algorithm to solve the special case of the problem optimally.

Define $f(j, x, q, u)$ to be the maximum objective value of a partial solution where (i) the first j coils have been considered (each is either added to a furnace or not); (ii) x furnaces have been used; (iii) u coils have been assigned to the current furnace (which is the x-th furnace); and (iv) q coils will finally be loaded into the current furnace. Define $G(j, q, u)$ to be the increase in objective value when coil j is assigned to a furnace which contains u coils before coil j is added and will have q coils in the end.

DP Algorithm:

Initial condition: $f(0, 0, 0, 0) = 0$.

Recursive relations: For $j = 1, \ldots, n$, $x = 1, \ldots, p$, $q = 1, \ldots, r$ and $u = 1, \ldots, q$:

$$f(j,x,q,u) = \begin{cases} \max\left\{ f(j-1,x,q,u),\ \max\{f(j-1,x-1,y,y) \mid 1 \le y \le r\} + G(j,q,0)\right\}, & \text{if } u = 1 \\ \max\left\{ f(j-1,x,q,u),\ f(j-1,x,q,u-1) + G(j,q,u-1)\right\}, & \text{if } u > 1 \end{cases}$$

$$\text{where}\quad G(j,q,u) = \begin{cases} F_j + \theta T_j, & \text{if } u \le \lfloor q/2 \rfloor \\ F_j - \theta T_j, & \text{if } u > \lceil q/2 \rceil \\ F_j, & \text{if } q \text{ is odd, and } u = \lceil q/2 \rceil \end{cases}$$

An optimal solution is found by calculating $\max\{f(n, x, u, u) \mid 1 \le x \le p,\ 1 \le u \le r\}$.

The DP considers two cases, $u = 1$ or $u > 1$, in calculating the value functions. In both cases, there are two ways of assigning coil j: it is either not assigned to any furnace, or assigned to the current furnace. In the former case, the objective value of the current state $f(j, x, q, u)$ remains the same as $f(j\text{-}1, x, q, u)$. In the latter case, coil j contributes $G(j, q, u\text{-}1)$ to the objective value $f(j, x, q, u)$. We note that when $u = 1$, the current furnace is a new furnace. In this case, if coil j is added to this new furnace, then the previous state that leads to the current state (j, x, q, u) must be in the form $(j\text{-}1, x\text{-}1, y, y)$, for some y that maximizes $f(j\text{-}1, x\text{-}1, y, y)$.

THEOREM 2: The above DP algorithm finds an optimal solution for the special case of the problem in $O(npr^2)$ time.

PROOF: See Appendix B.

5. Branch-and-Price-and-Cut Algorithm

We first reformulate the integer programming formulation BDP given in Section 3.2 for the general problem as a set-packing model in Section 5.1. We then propose a branch-and-price-and-cut solution algorithm to solve medium-size problem instances to optimality. The structure of the algorithm is the same as most branch-and-bound algorithms for integer programming problems where each branch-and-bound node (B&B node) is the LP relaxation of the original IP problem with additional constraints imposed by the branching rules used (Nemhauser and Wolsey 1988). We develop a combined column and row generation approach in Section 5.2 for solving the LP relaxation of the problem at each B&B node. This approach uses a column generation procedure with valid inequalities added into the LP relaxation in many iterations. A variable reduction strategy is proposed to prevent the generation of too many columns and, hence, accelerate the algorithm. Finally, in Section 5.3, we develop a layered branching strategy to obtain an optimal integer solution in the proposed branch and bound framework.

5.1. Reformulation

Let m be the number of different furnace types, where the furnaces of the same type are identical. Let p_j be the number of available furnaces of type $j \in \{1, \ldots, m\}$. We define S_j as the set of all feasible batching scenarios for a furnace of type j, and for each batching scenario $s \in S_j$, define c_j^s to be its *net reward value* (total reward of the coils in it minus total coil-coil and coil-furnace mismatching cost) and a_{ij}^s to be 1 if coil i is contained in batching scenario s and 0 otherwise. Define a decision variable λ_j^s to be 1 if batching scenario $s \in S_j$ is selected and 0, otherwise. The problem BDP can be reformulated as the following set packing model:

(SP-BDP) $$\text{Max } Z = \sum_{j=1}^{m} \sum_{s \in S_j} c_j^s \lambda_j^s \tag{9}$$

Subject to

$$\sum_{s \in S_j} \lambda_j^s \le p_j \qquad \text{for } j=1, \ldots, m \tag{10}$$

$$\sum_{j=1}^{m} \sum_{s \in S_j} a_{ij}^s \lambda_j^s \le 1 \qquad \text{for } i=1, \ldots, n \tag{11}$$

$$\lambda_j^s \in \{0,1\} \qquad \text{for } s \in S_j,\ j=1, \ldots, m \tag{12}$$

Constraint (10) requires that at most p_j feasible batching scenarios are assigned to furnaces of type j. Since in our problem there are more coils to be considered than the total capacity of the available furnaces, in an optimal solution, this constraint will be binding. We use inequality for this constraint instead of equality to make the column generation process more stable. Constraint (11) corresponds to constraint (2) ensuring that each coil can be assigned to at most one furnace. Let LSP-BDP denote the linear programming (LP) relaxation of SP-BDP. Then the solution of LSP-BDP provides an upper bound for SP-BDP.

To strengthen the LP relaxation, we add some valid inequalities to this formulation as follows. We say a coil i *dominates* another coil k if all of the following conditions are satisfied: (1) the mismatching cost between coil i and coil k is zero (i.e., these two coils have the same annealing curve, thickness, and outer-diameter), (2) the width of coil i is no more than that of coil k, (3) the reward value of i is larger than that of coil k (i.e., $F_i > F_k$), or the reward value of i is equal to that of k, but $i > k$. By this definition, it can be seen that if coil i dominates coil k, then coil i should be given a higher priority than coil k when selecting coils to form batches (i.e., if coil k is selected, then coil i should be selected too). For each coil i, define the *domination set* D_i to be the set of coils dominated by coil i. Let N_D be the set of coils with a non-empty domination set.

We construct a directed graph $G(N_D)$ to represent the domination relations between all possible pairs of coils as follows. For each pair of coils (i, j), where $i \in N_D$ and $j \in D_i$, we create a vertex i and a vertex j if they have not been created yet, and create a directed arc from vertex i to vertex j. Clearly, the resulting graph is a directed acyclic graph which is either a connected graph or a disconnected graph consisting of multiple disjoint subgraphs each of which is connected. We replace the entire graph $G(N_D)$ if it is connected or each connected subgraph of $G(N_D)$ by its unique transitive reduction (e.g. Aho *et al*. 1972). The *transitive reduction* G_r of a given directed acyclic graph G contains the same vertices as in G but a minimum subset of arcs in G such that there is a directed path from vertex i to vertex j in G if and only if there is a directed path from i to j in G_r. The resulting graph, denoted as $G_r(N_D)$, has fewer arcs than the original graph $G(N_D)$, but is sufficient for us to describe all the valid inequalities implied by the domination relations between the coils.

We add the following inequalities implied by the domination relations of the coils to LSP-BDP:

$$\sum_{j=1}^{m}\sum_{s\in S_j} a_{uj}^{s}\lambda_j^{s} \geq \sum_{j=1}^{m}\sum_{s\in S_j} a_{vj}^{s}\lambda_j^{s} \qquad \text{for each directed arc } (u, v)\in G_r(N_D) \qquad (13)$$

Using the reduced graph $G_r(N_D)$ instead of $G(N_D)$ prevents the other inequalities that are implied by the ones included in (13) from being included and hence removes some redundancies. Our computational experiment shows that for the problems with 100 coils we tested, on average, the reduced graph $G_r(N_D)$ has about a dozen arcs (i.e., there are about a dozen inequalities in (13)).

Due to the huge number of feasible batching scenarios, it would be impractical to enumerate all of them in solving the LP relaxation problem LSP-BDP. Therefore, a column generation technique is applied to solve LSP-BDP by solving a series of smaller problems that only include a limited number of feasible batching scenarios. Such a problem with only a limited number of feasible batching scenarios is called a restricted LP master problem, RLSP-BDP.

5.2. Column and Row Generation for LP Relaxation

Column generation is an effective method to solve linear programming (LP) problems with a large number of columns. It is often used within a branch and bound framework to solve large integer programs (Barnhart *et al*. 1998), including many combinatorial optimization problems (Lubbecke and Desrosiers 2005). The structure of the column generation algorithm for our problem is as follows. In each iteration, a restricted master problem containing only a subset of the columns of the LP relaxation LSP-BDP is solved first (by LP solver of CPLEX). The values of the dual variables obtained from solving the restricted master problem are used to update the pricing subproblems, which are then solved to find columns with a positive reduced cost in the maximization problem. If such columns exist, they are added to the restricted master problem. The procedure is repeated until there is no column with a positive reduced cost. The final solution obtained is the optimal solution for the LP relaxation. We also propose some valid inequalities and add them to the LP relaxation along with the columns generated to strengthen the LP relaxation. In addition, a variable reduction strategy is proposed to reduce the number of columns and speed up the algorithm.

Below we describe the valid inequalities, the pricing subproblems and the variable reduction strategy in Sections 5.2.1, 5.2.2 and 5.2.3, respectively.

5.2.1 Valid Inequalities

To strengthen the upper bound generated by solving the LP relaxation by the column generation approach, we propose some valid inequalities to tighten the solution space of the LP relaxation. Before giving the inequalities, we first introduce some definitions. Given the optimal solution of a LP relaxation problem, we define Π as the set of batching scenarios contained in the LP relaxation problem with a positive solution value, λ^e as the solution value of each batching scenario $e\in\Pi$, and Ω as the set of the coils contained in the batching scenarios in Π. For any three-coil subset $\Gamma=\{i_1, i_2, i_3\}$ of Ω, we define the *coverage value* of Γ as $\sum_{e\in\Pi(\Gamma)} \lambda^e$, where $\Pi(\Gamma)=\{ e\in\Pi \mid$ batching scenario e contains at least two of the three coils i_1, i_2, i_3 in $\Gamma\}$. We call a three-coil subset of Ω a *conflictual subset* if its coverage value is greater than 1. If the given solution of a LP relaxation is integral, then the coverage value of any three-coil subset Γ of Ω must be 0 or 1 and there is no conflictual three-coil subset. However, when the solution of the LP relaxation is fractional, this property may not hold as shown in the following example. As a part of the optimal solution to the LP relaxation problem LSP-BDP, for example, there are three batching scenarios e_1 consisting of coils 1 and 2, e_2 consisting of coils 2 and 3, and e_3 consisting of coils 1 and 3, and the solution value of each of these three batching scenarios is 1/2. The coverage value of the three-coil subset {1, 2, 3} is 3/2 (and hence {1, 2, 3} is a conflictual subset), while in any integral solution this value must be at most 1.

Given a fractional solution of a LP relaxation problem, with the coil set Ω defined as above, find every conflictual three-coil subset of Ω. Let v denote the number of such three-coil subsets found, and $\Gamma_{u+1}, \Gamma_{u+2}, \ldots, \Gamma_{u+v}$ denote these subsets, where u is the number of conflictual three-coil subsets found in earlier iterations. We can create the following valid inequalities corresponding to each conflictual three-coil subset Γ_l, as follows:

$$\sum_{j=1}^{m}\sum_{s\in S_j} t_l^s \lambda^s \le 1 \qquad \text{for } l = u+1, \ldots, u+v, \tag{14}$$

where t_l^s is 1 if batching scenario *s* contains at least two coils in Γ_l, and 0 otherwise.

The proposed valid inequality (14) for each given conflictual three-coil subset Γ_l can be viewed as a clique inequality (Nemhauser and Wolsey 1988) if our problem SP-BDP is viewed as an optimization problem in the following graph: view each batching scenario as a node, create an arc between each node containing at least two coils of Γ_l and each other node also containing at least two coils of Γ_l. It can be seen that all the batching scenarios that contain at least two coils of Γ_l form a clique in the graph, and at most one of these batching scenarios can be selected. There are studies in the literature that incorporate clique inequalities in solving difficult combinatorial optimization problems such as timetabling and vehicle routing problems (Avella and Vasil'Ev 2005, Baldacci *et al*. 2008).

In the process of solving the LP relaxation problem at a B&B node, we first apply the column generation procedure. When no column with a positive reduced cost can be generated, we perform the following row generation procedure: Find the coil set Ω and all three-coil conflictual subsets of Ω, generate a valid inequality (14) for each such three-coil subset, and add all these valid inequalities to the LP relaxation. Then the column generation procedure is run again. This process is repeated until no new column or row can be generated.

5.2.2 Pricing Subproblems

In the column generation procedure, pricing subproblems are solved to find columns with the largest reduced cost. Suppose that we have solved the LP relaxation of a restricted master problem containing *L* valid inequalities of the form (14), each corresponding to a conflictual three-coil subset Γ_l. Let σ_j, j=1, …, *m*, and π_i, i=1, …, *n*, denote the dual variables corresponding to constraints (10) and (11) respectively, β_l denote the dual variable value corresponding to the *l*-th valid inequality (14) (corresponding to Γ_l), and γ_{uv}, $(u, v) \in G_r(N_D)$, denote the dual variables corresponding to constraint (13). Let N_{jk} denote the set of coils which can be loaded into a furnace of type *j* with the median coil *k*, following the equipment and matching constraints described in Section 3.1. For a given furnace type $j = 1, 2, \ldots, m$, and a given median coil $k \in O_j$, the *pricing subproblem*, denoted as SUB(*j*, *k*), of finding a batching scenario $s \in S_j$ with the given median coil *k* that has the largest reduced cost can be formulated as the following binary integer program.

(SUB(*j*,*k*)) Max $$\delta_{jk} = \sum_{i \in N_{jk}} (F_i - C1_{ij} - C2_{ik} - \pi_i - \sum_{v \in D_i} \gamma_{iv} + \sum_{u \in \{u | i \in D_u\}} \gamma_{ui}) x_i - \sigma_j - \sum_{l=1}^{L} \beta_l z_l \qquad (15)$$

Subject to

$$\sum_{i \in N_{jk}} h_i x_i \le H - h_k \qquad (16)$$

$$z_l \ge x_i + x_h - 1 \qquad \text{for } i, h \in \Gamma_l,\ i > h;\ \text{and } l=1, \ldots, L \qquad (17)$$

$$x_i \in \{0,1\} \qquad \text{for } i \in N_{jk} \qquad (18)$$

$$z_l \in \{0,1\} \qquad \text{for } l=1, \ldots, L \qquad (19)$$

where x_i is a binary variable taking value 1 if the batching scenario to be found contains coil *i*, and 0 otherwise, and z_l is a binary variable taking value 1 if the batching scenario to be found contains at least two of the three coils in Γ_l, and 0 otherwise. Constraint (16) is the height capacity constraint corresponding to constraint (3). Constraint (17) defines the value of z_l. If the optimal objective value δ_{jk} is positive, then the column corresponding to the optimal solution is added into the restricted master problem. If for every furnace type $j = 1, 2, \ldots, m$ and median coil $k \in N$, solving the corresponding subproblem SUB(*j*, *k*) does not generate a column with a positive reduced cost, then the column generation procedure is terminated.

Each pricing subproblem SUB(*j*, *k*) is a generalization of the knapsack problem with the added complication that the objective value is reduced by a certain amount β_l whenever at least two of the three coils from a conflictual three-coil subset Γ_l are included in the final batching scenario. It is known that the knapsack problem is ordinarily NP-hard and can be solved to optimality by a pseudo-polynomial time algorithm (Kellerer *et al*. 2004). We show that the pricing subproblems are more difficult than the knapsack problem.

THEOREM 3: The pricing subproblems are strongly NP-hard.

PROOF: See Appendix B.

By Theorem 3, no pseudo-polynomial time algorithms exist for the pricing subproblems unless P=NP. To solve these problems efficiently, we develop a valid inequality as follows and solve the resulting formulation via the MIP solver of CPLEX. Let N_{jk}^r denote the subset of coils in N_{jk} that are not involved in valid inequalities (17) (i.e. not in any Γ_l, for $l = 1, \ldots, L$). Let SUB^r(*j*, *k*) denote the revised subproblem SUB(*j*, *k*) where the coil set N_{jk} is replaced by N_{jk}^r and all the parts related to z_l variables (i.e. the term $\sum_{l=1}^{L} \beta_l z_l$ in the objective function and constraints (17)) are removed from the formulation. Clearly, SUB^r(*j*, *k*) is the knapsack problem which can be solved optimally by a pseudo-polynomial time dynamic programming algorithm (Kellerer *et al*. 2004). The optimal objective value of SUB^r(*j*, *k*) is denoted as δ_{jk}^r. It is clear that the optimal objective value of SUB(*j*, *k*) must be at least δ_{jk}^r. This means that we can add the following valid inequality to SUB(*j*, *k*) when solving SUB(*j*, *k*):

$$\sum_{i \in N_{jk}} (F_i - \mathrm{C1}_{ij} - C2_{ik} - \pi_i - \sum_{v \in D_i} \gamma_{iv} + \sum_{u \in \{u | i \in D_u\}} \gamma_{ui}) x_i - \sigma_j - \sum_{l=1}^{L} \beta_l z_l \ge \delta_{jk}^r \qquad (20)$$

Since usually N_{jk}^r contains less than half of the coils in N_{jk}, an optimal solution of SUB^r(*j*, *k*) can be obtained in a very short amount of time. As shown later in our computational tests, the addition of the proposed valid inequality (20) can significantly reduce the running time of the MIP solver of CPLEX in solving the pricing subproblems involved in large size test problems.

5.2.3 Variable Reduction

In the process of column and row generation, a large number of columns may have to be generated. However, many of the columns generated are not part of the optimal solution. The presence of all these columns can slow down the algorithm. To accelerate the algorithm, we propose a variable reduction strategy to reduce the number of columns that have to be considered. The idea of our variable reduction strategy is similar to that used by Bianco *et al*. (1994) and Hadjar *et al*. (2006) for vehicle scheduling problems.

Given a B&B node, we denote $\xi^*=(\sigma, \pi, \beta, \gamma)$ and Z^{up}, respectively, as an optimal dual solution and the optimal objective value of the underlying LP relaxation problem, δ^* as the vector of the reduced costs with respect to the optimal dual solution ξ^*, δ^*_{jk} as the optimal objective value of the pricing subproblem SUB(j, k) corresponding to the dual solution ξ^*, and Z^{lo} as the objective value of a feasible solution of problem SP-BDP. Then we have the following Lemma.

LEMMA 2. If $\delta^*_{jk} < Z^{lo} - Z^{up}$, for some coil k and furnace j, then any optimal solution of problem SP-BDP cannot contain any batching scenario for furnace j where coil k is the median coil.
PROOF: See Appendix B.

If the condition in Lemma 2 is satisfied at any B&B node, batching scenarios for furnace j with median coil k should not be considered in its subsequent B&B nodes. This means that in any child node of this B&B node, we should not consider the pricing subproblem SUB(j, k) for any (j, k) that satisfies this lemma.

5.3. Branching Strategy

In our algorithm, the LP relaxation problem at each node of the branch-and-bound tree is solved by the column and row generation approach described in Section 5.2. If the optimal objective value of the LP relaxation problem is less than or equal to that of the incumbent solution, the node is pruned. Otherwise, we check if the solution is fractional or integral. If the solution is integral, then the corresponding B&B node is pruned. In this case, if the objective value is larger than the incumbent solution, then we replace the incumbent solution by this solution. Next we select a fractional node to branch on as follows. If the current B&B node is not pruned, then the depth-first-search rule is applied such that the current B&B node is selected as the next node to branch on. If the current B&B node is pruned, then the best-upper-bound rule is applied such that an active node in the B&B tree with the largest upper bound is selected to branch on.

It is crucial to design an effective branching strategy that exploits the problem structure. We propose the following branching strategy. Given the fractional LP relaxation solution λ of a B&B node being considered, we define the following parameters for all $i, q \in N$,

$$V_i = \sum_{j=1}^{m} \sum_{s \in S_j} a_{ij}^s \lambda_j^s \quad \text{and} \quad \theta_{iq} = \sum_{j=1}^{m} \sum_{s \in S_j} a_{ij}^s a_{qj}^s \lambda_j^s$$

The value of V_i represents the fraction of coil i contained in the furnaces, and this value should be 0 or 1 in any feasible integer solution. The value of θ_{iq} represents the fraction of both coils i and q contained in the same furnace, and this value should be 0 or 1 in any feasible integer solution.

LEMMA 3: If the solution of the LP relaxation problem of a B&B node is fractional, then there exist some coils i and q such that the θ_{iq} value is fractional.

PROOF: See Appendix B.

Lemma 3 implies that when all θ_{iq} variables are integral, the corresponding LP relaxation solution of the B&B node being considered must be integral.

Given the fractional LP relaxation solution λ of a B&B node being considered, we first branch on V_i values. If there are coils with fractional V_i values, we then select a coil i with V_i value closest to 0.5 and create two child nodes as follows. For the left child node, fix the value of V_i to be 0 by excluding coil i from being selected in the optimal solution. For the right child node, the value of V_i is fixed to be 1 by requiring that coil i be selected in the optimal solution. The master problem and pricing subproblems for each of these newly created nodes are created accordingly.

If there is no coil with a fractional V_i value, then we select a pair of coils i and q with θ_{iq} value closest to 0.5 and create two child nodes as follows. For the left child node, fix the variable θ_{iq} to be 0 by excluding batches containing both coils i and q from being selected in the optimal solution. The corresponding initial master problem consists of all the columns of its parent node except the batching scenarios that contain both coils i and q. At the same time, the constraint that $x_i + x_q \le 1$ is added to all pricing subproblems to guarantee that no batching scenarios that contain both coils i and q are generated. For the right child node, the variable θ_{iq} is fixed to be 1 by requiring that a batch containing both coils i and q is selected in the optimal solution. The corresponding initial master problem consists of all the columns of its parent node except the batching scenarios that contain exactly one of coils i and q. The constraint that $x_i = x_q$ is added to every pricing subproblems to ensure that any generated batching scenario contains either both coils i and q or none of them.

6. Tabu Search Heuristic

As discussed in Section 2, our problem is strongly NP-hard. Furthermore, there is no polynomial-time algorithm for this problem with a constant worst-case bound. The branch-and-price-and-cut algorithm proposed in Section 5 can only solve problems with a medium size. Therefore, we propose a tabu search heuristic to find near optimal solutions for large-scale problems. First, a greedy heuristic is proposed to obtain an initial solution. Then, three types of basic search neighborhoods are searched repeatedly to improve the solution. These basic search neighborhoods are defined as the set of solutions that can be obtained by an exchange between two coils in different furnaces, between one coil in a furnace and one coil not in any furnace, or between one coil in a furnace and two coils not in any furnace. If a solution obtained by applying a move is in the tabu list, then the move is constrained. When the solution cannot be improved any more, a sophisticated *variable depth neighborhood* is adopted to further improve the solution. Since the procedures for generating an initial solution and for constructing and searching the three basic neighborhoods are rather standard, we describe them in Appendix C, along with the step-by-step procedure of the tabu search heuristic.

In the remainder of this section, we briefly describe the construction of a variable depth neighborhood and how this neighborhood is searched. A variable depth neighborhood consists of a sequence of k simple moves where each move is performed on the solution obtained by its previous move and the depth k is dynamically determined. The purpose of using a variable depth neighborhood is to balance the scale of neighborhood and computational effort. An effective approach to explore a variable depth neighborhood is called *filter-and-fan method*. The method was initially proposed by Glover (1998) to refine solutions in scatter search. In recent years, many researchers have used the method to explore large-scale neighborhoods for solving complex problems including the job shop scheduling problem (Rego and Duarte 2009), the protein folding problem (Rego *et al*. 2011), and the resource-constrained project scheduling problem (Ranjbar 2008). The method can be illustrated as constructing a search tree where each node is a solution and each branch represents a basic move. In this paper, the implementation of the filter-and-fan method is slightly different from the existing literature.

We set the incumbent solution as the root node S_0 of the search tree. Basic neighborhoods of the root node are searched and the best η_1 solutions with the largest objective values are selected as nodes in the first level of the search tree, which are denoted as ($S_1(1)$, $S_1(2)$,…, $S_1(\eta_1)$). For $k \geq 1$, given the nodes generated in level k, the nodes in the next level k+1 are generated as follows. In level k, select η_1 nodes with the largest objective values and denote them as $S_k(1)$, $S_k(2)$,…, $S_k(\eta_1)$. For each selected node $S_k(l)$, l = 1, …, η_1, search its basic neighborhoods to generate many solutions and select the η_2 solutions with the largest objective values as part of the nodes in the next level k+1. This results in $\eta_1 \times \eta_2$ solutions in level k+1. The procedure terminates when a pre-specified maximum number of (e.g. 7) levels is reached or a node whose objective value is larger than that of the incumbent solution is obtained. The node with the maximum objective value in the search tree is set to be the current solution for the next round of tabu search. The detailed procedure of the proposed filter-and-fan method for the problem is given in Appendix C.

7. Computational Experiments

As discussed in Section 1, the number of coils to be annealed and the number of available furnaces in each shift vary from plant to plant, and within the same plant, vary from month to month and shift to shift. In practice, typically, the batching decision problem with no more than 100 coils in a shift is considered as medium-sized and the problem with more than 100 coils in a shift is considered as large sized. The largest possible problem instances can have up to 300 coils in a shift.

Our extensive computational experiments show that the branch-and-price-and-cut algorithm is capable of solving medium-sized problem instances to optimality within a time acceptable to a steel plant in practice. However, when the number of coils to be annealed is more than 100, the branch-and-price-and-cut algorithm could take a time longer than desired, and hence the tabu search heuristic should be used. Accordingly, to give a fair evaluation of these algorithms, we report the results from the following experiments that we conducted:

(i) Test the performance of the branch-and-price-and-cut algorithm using both randomly generated problem instances and real problem instances from Baosteel with a medium size.

(ii) Test the performance of the tabu search heuristic using real problem instances from Baosteel with a large size.

Baosteel previously used a rule-based planning approach (described in Section 7.1) to make coil batching decisions. Since many steel plants still use similar rule-based approaches, we use Baosteel's approach as a benchmark and compare it with our branch-and-price-and-cut algorithm and tabu search algorithm described in Sections 5 and 6 in the computational experiments. We collaborated with Baosteel in the last several years and developed a decision support system which replaced Baosteel's rule-based approach by our algorithms. The associated economic benefits for Baosteel are described in Section 8.

All the algorithms are executed on a PC with P4-3.0 GHz CPU and 1 GB memory. Baosteel's rule-based approach is given in Section 7.1. Computational results are reported in Sections 7.2 and 7.3.

7.1. A Typical Rule-Based Planning Approach

Most steel plants make coil batching decisions in the batch annealing process based on greedy rules and planners' experience. We briefly describe below the rule-based planning approach used by Baosteel prior to our collaboration. A flowchart of the approach is given in Appendix D. We believe that many other steel companies still use a similar approach.

In each iteration of their approach, planners at Baosteel first select an available furnace from the furnace type with the minimum number of remaining furnaces. This is to ensure that a maximum number of different furnace types is used, resulting in an overall maximum number of coil-furnace matching opportunities. Given a furnace, they then select a median coil and a batch of other coils to be loaded into this furnace, considering the matching requirements and constraints, as follows. According to the management requirements described in Section 3.1.3, a coil that has the highest priority index PRI (with ties broken by choosing the coil with the largest weight) and matches with the protective gas atmosphere of the furnace is selected as the median coil for the furnace. Select the coils which satisfy the following three conditions as the candidate coils to be put in the furnace together with the median coil: (1) annealing curve index is appropriate for the furnace, (2) annealing curve index is in the same subset as that of the median coil, and (3) outer-diameter difference and thickness difference with the median coil are not greater than the given thresholds defined by the planners. At first, the threshold for outer-diameter difference and that for thickness difference are set to be some small values respectively. If there are not enough coils satisfying the above conditions that can fill up the furnace, increase the threshold for outer-diameter difference and that for thickness difference and re-select the candidate coils satisfying the conditions until enough candidate coils are selected. Among the candidate coils identified, sequentially select the coils with the highest priority indices (with ties broken by weight) that can fill up the current furnace, and assign these coils to the furnace. Record the furnace and the coils assigned to it, and repeat the above process to generate a batching plan for next furnace.

The rule-based approach selects coils for inclusion in a furnace mainly based on priority values and weights. This approach has several obvious shortcomings: (i) it does not consider the coil widths in selecting coils for a furnace; (ii) it does not select median coils and other coils jointly; and (iii) it considers coils batch-by-batch and is greedy in nature.

7.2. Performance of the Branch-and-Price-and-Cut Algorithm

We first evaluate the performance of our branch-and-price-and-cut algorithm described in Section 5 by comparing it with the MIP Solver of CPLEX version 12.4 directly applied to the formulation BDP using randomly generated problem instances with a medium size. To test the effectiveness of the various special techniques developed in Section 5 (including the valid inequalities of Section 5.2.1, the variable reduction strategy of Section 5.2.3, and the valid inequality of Section 5.2.2 for the subproblems), we consider four versions of our algorithm as follows: (i) BP: the most basic version of the algorithm without using any of these special techniques, (ii) BPC: BP with the valid inequalities of Section 5.2.1, (iii) BPCR: BP with both the valid inequalities of Section 5.2.1 and the variable reduction strategy of Section 5.2.3, and (iv) BPCRC: BPCR with the subproblems solved using the valid inequality developed in Section 5.2.2.

We follow the structure of the real data from Baosteel to generate four sets of random test problems with a wide range of sizes that are viewed as medium in practice. The first set of problems (denoted as S1) all have 40 coils and 4 furnaces; the second set (denoted as S2) 60 coils and 6 furnaces; the third set (denoted as S3) 80 coils and 8 furnaces; and the fourth set (denoted as S4) 100 coils and 10 furnaces. There are four types of furnaces involved in all the test problems, representing the most commonly seen scenario in practice. Each set consists of 10 problems with the given numbers of coils and furnaces which are generated following the structure of the real production data collected from Baosteel (see Appendix E for a detailed description of test problem generation). The average results over the 10 test instances of each problem set are presented in Table 1. We set a computation time limit of two hours (7200 seconds) for each solution method. If an instance could not be solved to optimality within this time limit, we count its computation time as 7200 seconds.

Table 1. Comparing four versions of the branch-and-price-and-cut algorithm and the MIP Solver of CPLEX

Problem Sets	Method	Gap (%)	Zero gap	Node no.	Solved to opt.	Time (sec.)	Time red. r.t. CPLEX (%)	Solved to opt. by CPLEX
S1	BP	0.53	2	31	10	44.17	-6034.72%	10
	BPC	0.11	6	12.8	10	19.42	-2597.22%	10
	BPCR	0.11	6	12.6	10	5.75	-698.61%	10
	BPCRC	0.11	6	11.4	10	5.09	-606.94%	10
S2	BP	0.32	5	56	10	207.77	74.82%	9
	BPC	0.05	8	8.2	10	36.02	95.63%	9
	BPCR	0.05	8	8.2	10	10.23	98.76%	9
	BPCRC	0.05	8	8.2	10	11.68	98.58%	9
S3	BP	0.65	2	466	8	2247.96	37.71%	5
	BPC	0.23	4	309	9	1228.76	65.95%	5
	BPCR	0.23	4	316	10	283.38	92.15%	5
	BPCRC	0.23	4	302.8	10	260.88	92.77%	5
S4	BP	0.35	2	773.8	4	4350.01	-	0
	BPC	0.07	4	219.5	8	1812.52	-	0
	BPCR	0.07	4	277.1	9	1019.16	-	0
	BPCRC	0.07	4	261.8	10	612.40	-	0

In Table 1, Column "Gap (%)" is the relative integrality gap between the optimal solution and the LP relaxation solution obtained at the B&B root node. Column "Zero gap" specifies the number of instances (out of 10) whose integrality gap is zero (i.e., LP relaxation solution is optimal for the integer problem). The next two columns represent the number of B&B nodes explored, and the number of instances (out of the given 10) that are solved optimally within 7200 seconds, respectively. Column "Time (sec.)" is the average computation time of the 10 instances in seconds. Column "Time red. r.t. CPLEX (%)" gives the average computation time reduction relative to the time used by the MIP solver of CPLEX, where "-" represents that CPLEX is not capable of obtaining a feasible solution for any test instance due to memory overflow. The last column is the number of instances that are solved to optimality by CPLEX within 7200 seconds. Note that the MIP solver of CPLEX used here incorporates a so-called warm start strategy; it starts with the solution generated by the tabu search heuristic described in Section 6 as an initial solution. This version of the solver is faster than the default version which is not given any initial solution to start with.

By observing the results, our first conclusion is that the introduction of the row generation procedure (i.e.

using BPC over BP) effectively tightens the upper bounds, significantly reduces the integrality gap, and hence reduces the B&B nodes that need to be explored and shortens the computation time. Our second conclusion is that the introduction of variable reduction (i.e. using BPCR over BPC) further reduces the computation time in a significant way. Our third conclusion is that the introduction of the valid inequality for solving the subproblems further speeds up the algorithm for large problem instances. From BPCR to BPCRC, the average computation time essentially remains the same for S1 and S2 problems, whereas it is reduced slightly for S3 problems and significantly for S4 problems. BPCRC is the only version that solves every test problem to optimality within 7200 seconds. Our last conclusion is that for problems with 60 or more coils (S2, S3, S4), our branch-and-price-and-cut algorithm is substantially more effective than directly solving the MIP formulation of the problem by the CPLEX MIP solver. CPLEX cannot handle any problem in set S4 due to memory overflow, whereas all four versions of our algorithm solve most of the same problems to optimality within the same time limit. Also, our algorithm takes much less time than CPLEX for the problems that are solved to optimality. CPLEX has an advantage over our algorithm for S1 problems because of the relatively small size of these problems.

Another experiment is conducted to compare our branch-and-price-and-cut algorithm BPCRC and the rule-based approach described in Section 7.1 (denoted as RULE). Twenty representative real production problems collected from Baosteel, each corresponding to a shift in August 2007, are used for the experiment. The first six columns of Table 2 show the indices and sizes of these test problems including the number of coils and the number of furnaces for each furnace type. The test results are shown in the remaining columns of Table 2. Column “Imp on obj value (%)” is the improvement of objective value achieved by BPCRC relative to RULE. The columns under the heading “Average charging weight” are the average charging weight of a furnace in the solution generated by RULE and BPCRC and the improvement of this measure achieved by BPCRC relative to RULE. Column “BPCRC Gap (%)” is integrality gap of the optimal LP relaxation objective value obtained at the B&B root node in BPCRC relative to the optimal objective value of the problem. The last two columns are the run time of BPCRC and that of CPLEX solving formulation BDP directly, where 7200 means that the problem is not solved to optimality within the time limit of 7200 seconds.

From these results, we can see that the objective value is significantly improved by our branch-and-price-and-cut algorithm over the rule-based method. Also, the average charging weight is improved. We note that in practice, even a small improvement of the charging weight can mean a large profit increase due to increased productivity. The exact benefits brought to Baosteel by replacing their rule-based approach with our algorithms are estimated in Section 8. Furthermore, our branch-and-price-and-cut algorithm can obtain an optimal solution for every test problem within 25 minutes, whereas CPLEX is much less stable and fails to find an optimal solution in 2 hours for 4 test problems. In practice, most steel plants usually reserve thirty to sixty minutes to make a batching plan before each shift. So it is safe to choose the branch-and-price-and-cut algorithm to solve the batching decision problem with a medium size in practice.

Table 2. Comparing the branch-and-price-and-cut algorithm and the rule-based approach

Problem Index	Number of coils	Number of furnaces				Imp on obj value (%)	Average charging weight			BPCRC Gap (%)	Run time (s)	
		NH-big	NH-small	HH-big	HH-small		RULE	BPCRC	Imp (%)		BPCRC	CPLEX
1	54	2	1	1	1	2.85	86.20	87.38	1.37	0	3.53	30.58
2	52	2	1	1	1	0.87	87.75	88.61	0.98	0	8.83	9.14
3	40	2	1	1	1	6.22	84.08	84.77	0.82	0	16.32	4.92
4	56	2	1	1	1	14.89	88.23	89.99	1.99	0	6.71	7.05
5	65	2	1	1	1	0.64	87.08	87.78	0.80	0	10.65	9.53
6	77	3	2	2	2	15.70	88.07	90.78	3.08	0	21.87	7200
7	45	2	1	1	1	7.96	84.14	84.70	0.67	0	5.43	17.71
8	47	2	1	1	1	0.62	84.77	85.82	1.24	0.11	22.34	4.80
9	57	2	1	1	1	9.10	87.58	87.95	0.42	0.02	20.16	7200
10	79	3	2	2	2	6.31	86.85	87.35	0.58	1.41	313.53	7200
11	73	3	2	1	0	5.27	87.64	87.85	0.24	0	19.45	22.60
12	72	3	2	0	0	7.45	89.88	90.45	0.63	0.05	54.25	13.05
13	58	2	1	1	1	14.01	87.76	87.98	0.25	0.41	87.66	7.88
14	68	2	1	0	0	0.96	88.01	89.72	1.94	0	30.2	40.41
15	56	2	1	1	1	0.00	80.80	81.68	1.09	0	7.43	51.66
16	48	2	1	1	1	0.51	80.49	80.74	0.31	0.77	1258.37	966.75
17	65	2	1	1	1	17.36	83.95	85.24	1.54	0	6.97	28.64
18	51	2	1	1	1	23.47	74.38	75.64	1.69	0	4.95	8.91
19	56	2	1	1	1	8.76	86.71	86.81	0.12	0	6.54	21.74
20	75	3	2	2	2	33.60	84.18	89.83	6.71	0.06	172.36	7200
Average						8.83	85.43	86.55	1.32	0.14	103.88	1502.27

7.3. Performance of the Tabu Search Heuristic

In this section, we evaluate the performance of two versions of the tabu search heuristic by comparing them with our branch-and-price-and-cut algorithm as well as with the rule-based planning approach described in Section 7.1. The first version (denoted as BTABU) is the tabu search heuristic described in Section 6 using three basic neighborhoods only without using the variable depth neighborhood. The second version (denoted as VTABU) is the improved tabu search heuristic which incorporates the variable depth neighborhood.

We first compare the solutions obtained by BTABU and VTABU with the optimal solutions obtained by the branch-and-price-and-cut algorithm BPCRC using the same twenty medium size real problem instances from Baosteel given in Table 2. The test results are shown in Table 3 where column " Imp (%)" represents the relative improvement of objective value (or average charging weight) achieved by VTABU over BTABU, column "Gap (%)" is gap of the objective value (or average charging weight) given by VTABU relative to the optimal objective value obtained by BPCRC, and the columns under "Run time (s)" are the computational time taken by BTABU and VTABU, respectively.

Table 3. Comparing two versions of the tabu search heuristic and the branch-and-price-and-cut algorithm

Problem Index	Objective value		Average charging weight		Run time (s)	
	Imp(%)	Gap(%)	Imp(%)	Gap(%)	BTABU	VTABU
1	1.23	1.23	0.17	0.32	1.73	2.03
2	0.21	0.53	0.23	0.41	1.65	3.24
3	2.25	2.33	0.19	0.47	1.70	4.20
4	2.56	6.21	0.2	1.40	1.39	3.15
5	0.00	0.39	0.00	0.22	1.84	3.85
6	1.54	6.62	0.19	2.1	5.24	8.28
7	0.22	6.08	0.14	0.22	3.20	3.90
8	0.28	0.34	0.00	0.68	1.01	2.62
9	3.23	3.46	0.23	0.19	2.86	6.24
10	2.24	1.81	0.20	0.26	4.52	11.82
11	0.83	3.32	0.00	0.19	5.65	10.09
12	1.77	3.42	0.14	0.10	5.88	11.47
13	1.46	3.16	0.00	0.11	1.89	4.76
14	0.00	0.82	0.00	0.11	11.86	20.23
15	0.00	0	0.00	0.00	2.70	6.26
16	0.00	0.51	0.00	0.16	1.00	2.50
17	1.98	7.09	0.76	0.04	2.56	4.99
18	2.43	5.25	0.13	0.75	1.79	4.00
19	3.72	4.85	0.00	0.00	2.10	4.84
20	11.31	5.75	1.39	2.10	3.92	9.49
Average	1.86	3.16	0.20	0.49	3.22	6.40

Based on the results in Table 3, it can be seen that the tabu search heuristic can generate a near optimal solution for each problem tested in terms of both the overall objective value and the average charging weight achieved. In addition, it can be seen that VTABU generates slightly better solutions than BTABU, and both can solve every medium size problem tested within a few seconds.

Next we compare BTABU and VTABU with the rule-based approach (denoted as RULE) using twenty representative large size problem instances collected from Baosteel, each corresponding to the actual production data of a shift in October 2007. The first six columns of Table 4 show the indices and sizes of these problems including the number of available coils and the number of available furnaces for each furnace type. The test results are shown in the remaining columns of Table 4. Columns "Imp1 (%)" and "Imp2 (%)" represent the relative improvement of objective value achieved by VTABU over BTABU, and the relative improvement of objective value achieved by VTABU over RULE, respectively. Column "Imp3 (%)" is the relative improvement of charging weight achieved by VTABU over RULE. The columns under "Run time" are the computation times (in seconds) used by the corresponding methods. Column "Gap (%)" is the relative gap between the objective value given by VTABU and the upper bound (denoted as UB) obtained by solving the LP relaxation at the root node of the B&B tree, which is defined as $\frac{\text{UB-VTABU}}{\text{UB}} \times 100\%$.

Based on the results in Table 4, we can make the following observations about the tabu search heuristic. Compared to BTABU, VTABU improves the objective value by an average of 3.80%, which shows that the variable depth neighborhood is effective for avoiding the search being trapped and hence improving the solution. The average gap between the solution obtained by VTABU and the upper bound is 2.88%, which indicates that the proposed tabu search heuristic is capable of generating near optimal solutions. Furthermore, it takes the tabu search heuristic less than five minutes to generate a near optimal solution for each problem tested, whereas it could take more than one hour for the planners to generate a batching plan if it is done manually as in the case of some steel plants. Finally, we can see that the proposed tabu search heuristic makes a significant improvement over the rule-based approach in all dimensions including the objective value and the average charging weight. Improvement on the objective value represents improvement on the total reward of the selected coils and improvement of matching quality. Improvement on average charging weight translates into reduced production cost and increased productivity. The exact benefits brought to Baosteel by replacing their rule-based approach by our algorithms are estimated in Section 8.

Table 4. Comparing two versions of the tabu search heuristic and the rule-based approach

Problem Index	Number of coils	Number of furnaces				Objective value		Avg. charging weight			Run time (s)		Gap (%)
		NH-big	NH-small	HH-big	HH-small	Imp1(%)	Imp2(%)	RULE	VTABU	Imp3(%)	BTABU	VTABU	
1	210	8	5	5	2	0.24	13.44	89.35	91.20	2.07	60.49	108.37	4.65
2	205	7	5	6	3	1.47	24.24	88.15	92.61	5.06	73.98	80.44	1.09
3	113	4	2	2	1	19.60	18.39	87.99	88.58	0.67	15.23	35.40	0.87
4	226	7	5	5	4	6.17	9.67	91.53	93.18	1.80	54.95	130.70	1.00
5	114	4	4	0	2	10.86	8.81	85.00	86.82	2.14	10.20	28.29	2.29
6	175	6	6	1	5	0.90	12.57	85.61	92.03	7.50	27.95	96.09	3.07
7	145	5	6	1	3	0.69	11.25	86.91	92.71	6.67	50.50	78.87	2.70
8	107	4	3	0	1	2.45	7.10	81.19	81.37	0.22	16.78	29.17	4.63
9	162	7	5	2	1	1.69	4.70	84.79	85.27	0.57	28.78	81.73	4.04
10	201	5	8	4	6	6.65	17.20	86.04	86.26	0.26	82.14	105.79	6.01
11	189	9	7	2	5	0.70	13.37	84.84	86.61	2.09	146.36	188.62	1.69
12	216	10	5	1	5	0.07	7.46	89.96	90.63	0.74	98.16	150.20	2.06
13	190	8	4	3	4	1.41	19.57	84.49	86.19	2.01	70.27	121.46	1.37
14	132	7	7	1	0	5.16	12.44	86.72	87.18	0.53	109.09	131.06	0.93
15	88	3	2	2	0	5.63	2.87	87.92	88.38	0.52	10.03	30.50	3.28
16	145	6	4	1	2	0.57	4.38	86.63	87.05	0.48	32.07	96.48	3.71
17	135	4	4	2	3	5.98	6.73	91.28	92.17	0.98	32.55	89.75	3.96
18	87	5	3	1	0	1.25	4.90	80.39	82.19	2.24	10.22	23.34	3.35
19	120	6	4	1	1	4.53	8.52	94.10	95.76	1.76	19.39	40.26	4.78
20	220	8	6	3	5	0.24	16.42	83.41	83.93	0.62	134.06	241.03	2.13
Average						3.80	11.20	86.82	88.51	1.95	54.36	94.38	2.88

8. Benefits to Baosteel

A computerized batching decision support system (BDSS) for Baosteel that contains all the algorithms described in this paper was developed and implemented at Baosteel to replace their rule based approach (described in Section 7.1) with very satisfactory performance. A description of the BDSS is given in Appendix F. Below we estimate the annual economic benefits of the BDSS to Baosteel by generalizing the test results given in Sections 7.2 and 7.3 to the entire year such that the average results of Table 2 will be applied to all the low-demand months, and the average results of Table 4 will be applied to all the regular months. The demand for batch annealed coils that Baosteel experiences over time has a very similar pattern every year, which consists of 4 low-demand months and 8 regular months.

By Baosteel's statistics, in the low-demand seasons in the last 4 years, an average of 2160 batches of coils are annealed during the four low-demand months in a year. In the regular months of the same years, an average of 13680 batches of coils can be annealed during the eight regular months in a year. Generalizing the test results of Table 2, we can conclude that in the low-demand months, with the BDSS replacing the manual planning approach, the average charging weight of an annealing furnace is increased by 1.12 tons (from 85.43 tons to 86.55 tons). Similarly, generalizing the test results of Table 4, we can conclude that in the regular months, the use of the BDSS increases average charging weight of an annealing furnace by 1.69 tons (from 86.82 tons to 88.51 tons). As a result of the increase in the average charging weight brought by replacing the manual planning approach with the BDSS, the total annual tonnage of annealed coils is increased by 25538.4 tons (=1.12×2160+1.69×13680). By Baosteel's statistics, the per-ton net profit brought by batch annealing is US $68.93. Therefore, the increased annual tonnage of annealed coils due to the use of BDSS yields a net profit of about US $1.76 million (=68.93×25538.4) per year.

In addition, the increased average charging weight of a furnace due to the use of BDSS reduces the average per-ton consumption of coal gas, protective gas, electricity and water. Accordingly, the standard coal consumption and the carbon emission are also reduced. The use of the computerized system has also brought other benefits to Baosteel, which are hard to quantify but are equally significant to Baosteel. These benefits include improved quality of the annealed coils, improved customer satisfaction, and improved efficiency and ease of making batching decisions.

9. Conclusions

We have developed several efficient algorithms to assist steel plants in making better batching decisions to maximize productivity and energy utilization. We proposed a polynomial-time DP algorithm for a frequently occurring special case of the problem and designed a branch-and-price-and-cut exact algorithm that can solve medium size instances to optimality for the general case of the problem. Furthermore, a tabu search heuristic was designed which can solve large size instances to near optimality. Our computational experiments have demonstrated that our algorithms outperform the rule-based planning approach Baosteel previously used and many other steel plants may still be using. We have collaborated with Baosteel and replaced their rule-based planning approach by a decision support system that contains our algorithms. The use of our algorithms has brought significant tangible and intangible benefits to Baosteel.

Acknowledgements

The authors would like to thank the editor-in-chief Dr. Steve Graves, the associate editor and the two referees for their suggestions and comments which have enabled the authors to significantly improve the paper.

References

Aho, A.V., M.R. Garey, J.D. Ullman. 1972. The transitive reduction of a directed graph. *SIAM Journal on Computing*. **1** 131-137.

Avella, P., I. Vasil'Ev. 2005. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*. **8** 497-514.

Balakrishnan, A., J. Geunes. 2003. Production planning with flexible product specifications: An application to specialty steel manufacturing. *Operations Research*. **51**(1) 94-112.

Baldacci, R., N. Christofides, A. Mingozzi. 2008. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*. **115**(2) 351-385.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: column generation for solving huge integer programs. *Operations Research*. **46**(3) 316-329.

Bianco, L., A. Mingozzi, S. Ricciardelli. 1994. A set partitioning approach to the multiple depot vehicle scheduling problem. *Optimization Methods and Software*. **3** 163-194.

Chekuri, C., S. Khanna. 2006. A PTAS for the multiple knapsack problem. *SIAM Journal on Computing*. **35** 713-728.

Chu, C.B., J. Antonio. 1999. Approximation algorithms to solve real-life multicriteria cutting stock problems. *Operations Research*. **47**(4) 495-508.

Dawande, M., J. Kalagnanam, P. Keskinocak, P. Ravi, F. Salman. 2000. Approximation algorithms for the multiple knapsack problem with assignment restrictions. *Journal of Combinatorial Optimization*. **4**(2) 171-186.

Dutta, G., R. Fourer. 2001. A survey of mathematical programming applications in integrated steel plants. *Manufacturing & Service Operations Management*. **3**(4) 387-400.

Glover, F. 1998. A template for scatter search and path relinking, in: J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), Artificial Evolution, Lecture Notes in Computer Science, Springer, Heidelberg. **1363** 13-54.

Hadjar, A., O. Marcotte, F. Soumis. 2006. A branch-and-cut algorithm for the depot vehicle scheduling problem. Operations Research. **54**(1) 130-149.

Höhn, W., T. Jacobs, N. Megow. 2012. On Eulerian extension and their application to no-wait flowshop scheduling. *Journal of Scheduling*. **15**(3) 295-309.

Höhn, W., F.G. König, R.H. Möhring. 2011. Integrated sequencing and scheduling in coil coating. *Management Science*. **57**(4) 647-666.

Kalagnanam, J.R., M.W. Dawande, M. Trumbo, H.S. Lee. 2000. The surplus inventory matching problem in the process industry. *Operations Research*. **48**(4) 505-516.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer-Verlag Berlin.

König, F.G., M. Lübbecke, R. Möhring, G. Schäfer, I. Spenke. 2007. Solutions to real-world instances of PSPACE-complete stacking. *Algorithms*. **4698** 729-740.

Lee, C.-Y., R. Uzsoy, L.A. Martin-Vega. 1992. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*. **40** (4) 764-775.

Lin, J.-H., J.S. Vitter. 1992. ε-Approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771-782.

Lubbecke, M.E., J. Desrosiers. 2005. Selected topics in column generation. *Operations Research*. **53**(6) 1007 -1023.

Martello, S., P. Toth. 1990. Knapsack problems: algorithms and computer implementations. *John Wiley & Sons*.

Megiddo, N., J. Supowit. 1984. On the complexity of some common geometric location problems. *SIAM Journal on Computing*. **13** 182-196.

Möhring, R.H., S.S. Andreas, F. Stork, M. Uetz. 2003. Solving project scheduling problems by minimum cut computations. *Management Science*. **49**(3) 330-350.

Moon, S., A.N. Hrymak. 1999. Scheduling of the batch annealing process-deterministic case. *Computers and Chemical Engineering*. **23**(9) 1193-1208.

Naphade, K.S., S.D. Wu, R.H. Storer, B.J. Doshi. 2001. Melt scheduling to trade off material waste and shipping performance. *Operations Research*. **49**(5) 629-645.

Nemhauser, G.L., L.A. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons.

Prasad, P., T. Maravelias. 2008. Batch selection, assignment and sequencing in multi-stage multi-product processes. *Computers and Chemical Engineering*. **32** 1106-1119.

Ranjbar, M. 2008. Solving the resource-constrained project scheduling problem using filter-and-fan approach. *Applied Mathematics and Computation*. **201**(1-2) 313-318.

Rego, C., R. Duarte. 2009. A filter-and-fan approach to the job shop scheduling problem. *European Journal of Operational Research*. **194**(3) 650-662.

Rego, C., H. Li, F. Glover. 2011. A filter-and-fan approach to the 2D HP model of the protein folding problem. *Annals of Operations Research*. **188**(1) 389-414.

Tang, L.X., G. Wang. 2008. Decision support system for the batching problems of steelmaking and continuous-casting production. *Omega*. **36**(6) 976-991.

Tang, L.X., G. Wang, Z.-L. Chen. 2014. Integrated charge batching and casting width selection at Baosteel. *Operations Research*. **62**(4) 772-787.

Tang, L.X., G. Wang, J.Y. Liu. 2011. A combination of Lagrangian relaxation and column generation for order batching in steelmaking and continuous-casting production. *Naval Research Logistics*. **58**(4) 370-388.

Tang, L.X., J.Y. Liu, A.Y. Rong, Z.H. Yang. 2000. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*. **124**(2) 267-282.

Tang, L.X., J.Y. Liu, A.Y. Rong, Z.H. Yang. 2001. A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*. **133**(1) 1-20.

Tang, L.X., S.J. Jiang. 2009. The charge batching planning problem in steelmaking process using Lagrangian relaxation algorithm. *Industrial & Engineering Chemistry Research*. **48**(16) 7780-7787.

Tang, L.X., X. Xie, J.Y. Liu. 2009. Scheduling of a single crane in batch annealing process. *Computers & Operations Research*. **36**(10) 2853-2865.

Vonderembse, M.A., R.W. Haessler. 1982. A mathematical programming approach to schedule master slab casters in the steel industry. *Management Science*. **28**(12) 1450-1461.

All appendices are published as online supplement

Appendix A: Parameter Settings at Baosteel

1. F_i : *Reward value of coil i*

As described in Section 3.2, the reward value of coil i, F_i, is calculated as $\rho f_i+(1-\rho)g_i$, where ρ is the weight used for the PRI value f_i. In most cases, Baosteel uses $\rho = 0.5$ to reflect the fact that they view the PRI factor and the weight factor equally important in setting the reward value of a coil. Whenever necessary, Baosteel may change the value of ρ according to specific production and management requirements.

The PRI value of coil i, f_i, is mainly used to model the management requirements which are described in section 3. The management requirements consist of due dates, relationship with other coils, contract accumulation times and storage period of coils. Then the value of f_i can be calculated by $f_i = \alpha_1 P_i^1 + \alpha_2 P_i^2 + \alpha_3 P_i^3 + \alpha_4 P_i^4$, where α_1, α_2, α_3, α_4 are the PRI weights given by planners according to their management requirement, and P_i^1, P_i^2, P_i^3, and P_i^4 are used to represent the values of the corresponding PRI items, respectively, which are described as follows.

1) PRI item about the due date slack of a coil:

$$P_i^1 = \begin{cases} F1_1^1, & A_i < A \\ F1_1^2, & A \le A_i \le B \\ F1_1^3, & A_i > B \end{cases}$$

where $F1_1^2 > F1_1^1 > F1_1^3$. A_i is the time between the current date when the batching decision is being made and the due date of coil i. A and B are respectively the lower bound and upper bound of the total processing time for a coil. If $A_i \in [A, B]$, then coil i is the most urgent coil and should be given the highest priority for annealing to meet the required due date. If $A_i<A$, then coil i cannot be completed before its due data (i.e., it will be tardy) even if coil i is annealed immediately, and hence a lower priority is assigned to it. If $A_i>B$, then there is sufficient time for coil i to be completed before its due date, hence the coil should be given the lowest priority. In Baosteel, the values of A and B are set to be 5 days and 7 days, respectively.

2) PRI item about relationship with other coils:

$$P_i^2 = \max\begin{cases} F1_2^1, & \text{if the sorting grade of coil } i \text{ is 4} \\ F1_2^2, & Curv_i \in Spec\_C \\ F1_2^3, & Grad_i \in Spec\_G \\ F1_2^4, & Thick_i \le Low\_Th \\ 0, & \text{otherwise} \end{cases}$$

In Baosteel, sorting grade is used to identify the surface flatness level of a coil. The coils with sorting grade of 4 are required to be with the highest level of surface flatness and should be batched together in the finishing operation after batch annealing. Hence, to reduce the setup in the finishing operation, the coils whose sorting grades are 4 should be given a priority for annealing. Additionally, coil i, which satisfies any of the following three conditions, is difficult to match with other coils and need to be given a priority to avoid overstocking: (1) annealing curve index $Curv_i$ belongs to a special annealing curve index set $Spec\_C$, or (2) steel grade $Grad_i$ belongs to a special steel grade set $Spec\_G$, or (3) thickness $Thick_i$ is no larger than a threshold value Low_Th (i.e., 1.1mm in Baosteel). According to the actual production situation of Baosteel, the priority values for the above four cases are set in sequence of $F1_2^1 > F1_2^2 > F1_2^3 > F1_2^4$.

3) PRI item about contract accumulation times

$$P_i^3 = \begin{cases} F1_3^1, & Per_i \ge Low\_Per \\ 0, & \text{otherwise} \end{cases}$$

Per$_i$ is the total weight of the coils which belong to the same contract as coil *i* and have completed all production processes (and hence are waiting for coil *i* before being delivered together to their customer). When *Per*$_i$ is no less than a threshold value *Low_Per* (which is set to be 4 tons in Baosteel), it represents that there are a certain number of coils that have completed all production processes and are waiting for coil *i*. In this situation, coil *i* should be processed as quickly as possible to shorten the contract accumulation time.

4) PRI item about storage time of coils

$$P_i^4 = \begin{cases} F1_4^1, & d_i \geq D \\ 0, & d_i < D \end{cases}$$

Here d_i is the amount of time that coil *i* has spent in storage, and *D* is a threshold value (which is set to be 5 days in Baosteel). When d_i is larger than *D*, coil *i* should be processed as soon as possible.

2. *C_{1ij}: Mismatching penalty between coil i and furnace j*

C_{1ij} is the mismatching penalty between coil *i* and batch annealing furnace *j*. The matching constraints between coils and furnaces have been described in section 3.1. The coil whose annealing curve belongs to set ACS1= {01, 02, 04, 05, 11, 12, 13, 23} can be annealed in furnaces with either NH type or HH type, but if it is annealed in a furnace of HH type, there is a penalty. The coil, whose annealing curve belongs to ACS2= {61, 62, 63, 64, 65, 66, 67, 68}, can only be annealed in furnaces with HH type. Accordingly, the value of C_{1ij} is given as follows:

$$C_{1ij} = \begin{cases} 0, & \text{if the annealing curve of coil } i \text{ belongs to ACS1 and the type of furnace } j \text{ is NH} \\ 0, & \text{if the annealing curve of coil } i \text{ belongs to ACS2 and the type of furnace } j \text{ is HH} \\ F2_1^1, & \text{if the annealing curve of coil } i \text{ belongs to ACS1 and the type of furnace } j \text{ is HH} \end{cases}$$

3. *C_{2ik}: Mismatching penalty between coil i and coil k*

C_{2ik} is the penalty caused by the mismatching between coil *i* and coil *k*, considering three main factors - the annealing curve index, thickness, outer-diameter of the coils. The value of C_{2ik} is a weighted sum of three penalty terms reflecting these three factors, respectively, i.e., $C_{2ik}=\beta_1 C_{2ik}^1 + \beta_2 C_{2ik}^2 + \beta_3 C_{2ik}^3$, where β_1, β_2, β_3 are the penalty weights set by planners according the actual situation, and

$$C_{2ik}^1 = \begin{cases} 0, & \text{if the annealing curve index of coil } i \text{ is the same as that of coil } k \\ F3_1^1, & \text{if the annealing curve indices of coil } i \text{ and coil } k \text{ are different but in the same subset} \end{cases}$$

$$C_{2ik}^2 = \begin{cases} 0, & \text{if } |T_i\text{-}T_k| \leq TT \\ F3_2^1 \times |T_i\text{-}T_k|, & \text{otherwise} \end{cases}$$

where T_i and T_k are the thicknesses of coils *i* and *k*, respectively, and *TT* is a threshold of thickness difference (which is set to be 50mm by Baosteel).

$$C_{2ik}^3 = F3_3^1 \times | d_i - d_k |,$$

where d_i and d_k are the outer-diameters of coils *i* and *k*, respectively.

All the parameter values discussed above are set according to Baosteel's actual production and management requirements. Different values of the parameters described above may produce different batching plans. The planners have the flexibility to change them according to their preferred requirements for batching quality.

Appendix B: Proofs of Lemmas 1, 2 & 3, and Theorems 1, 2 and 3

LEMMA 1: For any given batching scenario in an optimal solution, reindex the coils in this furnace as [1], [2], …, [q], such that $T_{[1]} \geq T_{[2]} \geq \ldots \geq T_{[q]}$, where $q \leq r$ is the number of coils in the furnace. The median coil for this furnace must be coil [l], where $l = \lceil q/2 \rceil$.

PROOF: Given a batching scenario of q coils indexed as [1], [2], …, [q] such that $T_{[1]} \geq T_{[2]} \geq \ldots \geq T_{[q]}$, denote $f(k)$ to be the total mismatching cost between the median coil and all other coils in this batch if coil [k] is chosen as the median coil. Since $C_{2[i][k]} = \theta|T_{[i]} - T_{[k]}|$, we have,

$$f(k+1) - f(k) = \sum_{i=1}^{k} C_{2[i],[k+1]} + \sum_{i=k+2}^{q} C_{2[i],[k+1]} - \sum_{i=1}^{k-1} C_{2[i][k]} - \sum_{i=k+1}^{q} C_{2[i][k]}$$

$$= \sum_{i=1}^{k} \theta\left|T_{[i]} - T_{[k+1]}\right| + \sum_{i=k+2}^{q} \theta\left|T_{[k+1]} - T_{[i]}\right| - \sum_{i=1}^{k-1} \theta\left|T_{[i]} - T_{[k]}\right| - \sum_{i=k+1}^{q} \theta\left|T_{[k]} - T_{[i]}\right|$$

$$= \theta\sum_{i=1}^{k} T_{[i]} - k\theta T_{[k+1]} + \theta(q-k-1)T_{[k+1]} - \theta\sum_{i=k+2}^{q} T_{[i]} - \theta\sum_{i=1}^{k-1} T_{[i]} + (k-1)\theta T_{[k]}$$

$$- \theta(q-k)T_{[k]} + \theta\sum_{i=k+1}^{q} T_{[i]}$$

$$= \theta(2k-q)(T_{[k]} - T_{[k+1]})$$

We can see that, if $2k \geq q$, then $f(k)$ is a monotonically non-decreasing function, and if $2k \leq q$, then $f(k)$ is a monotonically non-increasing function. Therefore, $f(k)$ takes the minimum value when $k = l$, i.e., selecting coil [l] as the median coil minimizes the mismatching cost of the batch. □

THEOREM 1: There exists an optimal solution where the index range of all the coils assigned to any furnace is non-overlapping with that of all the coils assigned to any other furnace.

PROOF: We first prove the following claim which will be used to prove the theorem later:

Claim: There exists an optimal solution that satisfies the following conditions: (i) the index range of the left coils in the batching scenario of any furnace is non-overlapping with that of the left coils in the batching scenario of any other furnace, and (ii) the index range of the right coils in the batching scenario of any furnace is non-overlapping with that of the right coils in the batching scenario of any other furnace.

We prove this claim by showing that given a solution δ not satisfying the conditions stated in the above claim, we can always find a new solution δ' which satisfies the conditions and has a total net reward no smaller than that of δ.

Suppose that in the given solution δ, the conditions do not hold. Suppose that there are b ($b \leq P$) batches in the solution δ. We rearrange the coils in these batches as follows. First, we sequence these batches of coils in the non-decreasing order of the indices of their median coils, and denote these batches as B_1, B_2, …, B_b. Let α_j denote the median coil, L_j the set of the left coils and R_j the set of the right coils in B_j respectively, for $j = 1, \ldots, b$. Next, we re-assign the left coils and right coils to these batches, respectively as follows: take all the left (right) coils from all the batches, put them together and sequence them in the non-decreasing order of their indices. The resulting sequence of the left (right) coils is denoted as Γ_l (Γ_r). Then sequentially re-assign these coils back to the b batches as follows: assign the first $|L_1|$ ($|R_1|$) coils in Γ_l (Γ_r), denoted as set U_1 (V_1), to batch B_1; assign the next $|L_2|$ ($|R_2|$) coils in Γ_l (Γ_r), denoted as U_2 (V_2), to batch B_2; …., assign the last $|L_b|$ ($|R_b|$) coils in Γ_l (Γ_r), denoted as U_b (V_b), to the last batch B_b. This results in b new batches, denoted as B_1', B_2', …, B_b', where B_j' consists of U_j, α_j, and V_j, for $j = 1, \ldots, b$. A new feasible solution δ' consisting of these b new batches is obtained. Since, in any batch B_j, there are at least $\sum_{k=1}^{j}|L_k|$ coils in Γ_l whose indices are smaller than that of its median coil α_j in the new batch $B_j^{'}$, the index of coil α_j is greater than the index of any coil in U_j. Similarly, we can show that in $B_j^{'}$, the index of coil α_j is smaller than the index of any coil in V_j. This shows that α_j is the median coil, U_j is the set of left coils and V_j is the set of right coils in $B_j^{'}$. Clearly, in the new solution δ', the index ranges of all the left coils are non-overlapping, and the index ranges of all the right coils are also non-overlapping.

The total mismatching cost between the median coils and the other coils in solution δ, denoted as MC, is equal to $\theta\sum_{j=1}^{b}\left(\sum_{i\in L_j}(T_i - T_{\alpha_j}) + \sum_{i\in R_j}(T_{\alpha_j} - T_i)\right)$. Similarly, the total mismatching cost between the median coils and the other coils in solution δ', denoted as $MC^{'}$, is equal to $\theta\sum_{j=1}^{b}\left(\sum_{i\in U_j}(T_i - T_{\alpha_j}) + \sum_{i\in V_j}(T_{\alpha_j} - T_i)\right)$. By the fact that $|L_j| = |U_j|$ and $|R_j| = |V_j|$, for $j = 1, \ldots, b$, and $L_1 \cup L_2 \cup \ldots \cup L_b = U_1 \cup U_2 \cup \ldots \cup U_b$, and $R_1 \cup R_2 \cup \ldots \cup R_b = V_1 \cup V_2 \cup \ldots \cup V_b$, it is easy to see that $MC = MC^{'}$. Therefore, the total net revenue in δ is equal to that in solution $\delta^{'}$. This shows that the claim stated at the beginning of the proof holds.

Now we prove the theorem. Suppose that in an optimal solution δ, the index ranges of coils assigned to different furnaces overlap. Without loss of generality, we assume that there are b batches in δ, $B_1, B_2, \ldots, B_b$, which are in the increasing order of their indices of their median coils. By the claim proved earlier, the overlapping can only happen between the right-coils of a batch B_j and the left-coils of another batch $B_{j'}$ with a larger index (i.e., $j' > j$). Without loss of generality, assume i is a right coil in B_j, k is a left coil in $B_{j'}$ and $i > k$. It is not hard to see that exchanging i and k will not increase the mismatching cost in either B_j or an $B_{j'}$. As the total reward is unchanged, the above exchange will not reduce the objective value. Therefore there is an optimal solution satisfying the conditions in this theorem. □

THEOREM 2: The above DP algorithm finds an optimal solution for the special case of the problem in O(npr^2) time.

PROOF: The discussion given right after the description of the algorithm in the main body of the paper shows the correctness of the recursive relations and hence the optimality of the algorithm. The total computational time of the DP includes the time for calculating G function values and the time for calculating f function values. It is easy to see that it takes O(nr^2) time to calculate G values. To estimate the time for calculating f values, we consider two cases. When $u = 1$, there are a total of O(npr) states in the DP, and for each state, it takes O(r) time to calculate the value of the state. When $u > 1$, there are a total of O(npr^2) states in the DP, and for each state, it takes a constant time to calculate the value of the state. Therefore, it takes a total of O(npr^2) time to calculate f values. This shows that the time complexity of the DP is O(nr^2) + O(npr^2) = O(npr^2). □

THEOREM 3: The pricing subproblems are strongly NP-hard.

PROOF: We prove that each pricing subproblem SUB(j, k) is strongly NP-hard by a reduction from the strongly NP-complete Independent Set problem (Garey and Johnson 1979).

Independent Set (IS): Given a graph $G = (V, E)$ and a positive integer $K \le |V|$, does G contain an independent set of size K or more, i.e. does there exist a subset $U \subseteq V$ such that $|U| \ge K$ and such that no two vertices in U are joined by an edge in E?

Given an instance of the above described IS, we construct an instance of SUB(j, k) as follows. For ease of presentation, we denote $v = |V|$, $V = \{1, \ldots, v\}$, $w = |E|$, and $E = \{\varepsilon_1, \ldots, \varepsilon_w\}$. For each edge $\varepsilon_i \in E$, define $V(\varepsilon_i)$ as the set of the two vertices of the edge. Clearly, $V(\varepsilon_i)$ is a subset of V. In the pricing subproblem SUB(j, k), there are $v+w$ coils with the coil set $N_{jk} = \{1, \ldots, v, v+1, \ldots, v+w\}$, where each coil i, for $i = 1, \ldots, v$, corresponds to vertex $i \in V$, and coil $v + i$, for $i = 1, \ldots, w$, corresponds to edge $\varepsilon_i \in E$. Let $L = w$ be the number of conflictual three-coil subsets, where for $l = 1, \ldots, L$, the *l-th* conflictual three-coil subset consists of coil $v+l$ and the two coils corresponding to the two vertices of edge ε_l, i.e., $\Gamma_l = \{v+l\} \cup V(\varepsilon_l)$. The parameters in the objective function of SUB(j, k) are defined as follows: Let the parameter of variable x_i be 1 for every $i \in V$, and 0 for every $i \in N_{jk} \backslash V$. Let $\sigma_j = 0$. Let $\beta_l = 1$, for $l = 1, \ldots, L$. This the objective function (15) becomes: Max $\sum_{i\in V} x_i - \sum_{l=1}^{L} z_l$. The parameters in the constraints are defined as follows: $h_i = 1$ for every $i \in N_{jk}$ and $H = K$. The constraints in the problem consist of (16), (17) for each pair of coils in Γ_l, for $l = 1, \ldots, L$, (18) and (19). The threshold of the objective value is K.

We show below that there is a solution to the pricing subproblem SUB(j, k) with the objective value at least K if and only if there is a solution to the instance of IS.

<u>*"If"* Part:</u> Suppose that for the instance of IS, there exists a subset $U \subseteq V$ such that $|U| \geq K$ and such that no two vertices in U are joined by an edge in E. We construct a solution to the instance of SUB(j, k) as follows. Choose arbitrary K vertices from the subset U to form another subset I. Let $x_i = 1$ for $i \in I$, and $x_i = 0$ for $i \in N_{jk} \setminus I$. Let $z_l = 0$ for every $l = 1, \ldots, L$. It is clear that this solution has an objective value of K. We next show that this solution is feasible. Since exactly K of the x_i variables have a value of 1, constraint (16) is obviously satisfied. Consider any conflictual three-coil subset $\Gamma_l = \{i_1, i_2, i_3\}$ where $i_1 < i_2 < i_3$. Since $\Gamma_l = \{v+l\} \cup V(\varepsilon_l)$), we can conclude that $i_3 = v+l$, and i_1 and i_2 correspond to the two vertices of edge ε_l. Thus, $x_{i_3} = 0$. Also, since no two vertices in I are joined by an edge in E, we have: $x_{i_1} + x_{i_2} \leq 1$. Thus constraint (17) is satisfied for any pair of coils in Γ_l. This shows that the constructed solution is feasible for SUB(j, k).

<u>*"Only If"* Part:</u> Suppose that there is a feasible solution to the instance of SUB(j, k) with an objective value of at least K. By constraint (16), and the fact that the objective value of this solution is at least K, we know that in this solution, (i) exactly K coils of V have an x_i value 1, and all other coils have an x_i value 0 , and (ii) $z_l = 0$ for every $l = 1, \ldots, L$. Let U be the subset of V consisting of the K coils with an x_i value 1. Fact (ii) implies that $x_i + x_q \leq 1$ for any pair of coils i and q in Γ_l, for every $l = 1, \ldots, L$. This means that U contains at most one of the two vertices of ε_l, for every $l = 1, \ldots, L$. In other words, for each edge in E, at most one of the two vertices of this edge is contained in U. Thus U is a solution to the instance of IS. □

LEMMA 2. If $\delta^*_{jk} < Z^{lo} - Z^{up}$, for some coil k and furnace j, then any optimal solution of problem SP-BDP cannot contain any batching scenario for furnace j where coil k is the median coil.

PROOF: For ease of presentation, let ***c***, ***A*** and ***b*** denote, respectively, the objective coefficient vector in (9), the constraint coefficient matrix, and the right-hand-side coefficient vector corresponding to constraints (10), (11), (13) and (14) in problem SP-BDP. From the linear programming duality, we know that $Z^{up} = \boldsymbol{\xi}^*\boldsymbol{b}$ and $\boldsymbol{\delta}^* = \boldsymbol{c} - \boldsymbol{\xi}^*\boldsymbol{A}$. Suppose that there is an optimal solution of the integer problem SP-BDP, $\boldsymbol{\lambda}^*$, which contains a batching scenario s for furnace j where coil k is the median coil, i.e., $\lambda^{*s}_j = 1$.

From the feasibility of $\boldsymbol{\lambda}^*$ to the LP relaxation LSP-BDP, we can have the inequality $\boldsymbol{A\lambda}^* \leq \boldsymbol{b}$. Since $\boldsymbol{\xi}^*$ is nonnegative, the inequality $\boldsymbol{\xi}^*\boldsymbol{A\lambda}^* \leq \boldsymbol{\xi}^*\boldsymbol{b}$ can be easily obtained. Combining with the fact that $\boldsymbol{Z}^{lo} \leq \boldsymbol{c\lambda}^*$, we have: $Z^{lo} \leq (\boldsymbol{c} - \boldsymbol{\xi}^*\boldsymbol{A})\boldsymbol{\lambda}^* + \boldsymbol{\xi}^*\boldsymbol{b}$. Based on the definitions of Z^{up} and $\boldsymbol{\delta}^*$, the above inequality can be rewritten as $Z^{lo} \leq \boldsymbol{\delta}^*\boldsymbol{\lambda}^* + Z^{up}$. When $\lambda^{*s}_j = 1$, the non-negativity of $\boldsymbol{\lambda}^*$ and the non-positivity of the reduced cost vector imply that $Z^{lo} \leq \delta^{*s}_j \lambda^{*s}_j + Z^{up} = \delta^{*s}_j + Z^{up}$. Then, we can get $\delta^{*s}_j \geq Z^{lo} - Z^{up}$. By the definition of δ^*_{jk}, the relationship $\delta^*_{jk} \geq \delta^{*s}_j$ can easily be obtained and hence the inequality $\delta^*_{jk} \geq Z^{lo} - Z^{up}$ can also be easily deduced. This contradicts with the fact that $\delta^*_{jk} < Z^{lo} - Z^{up}$. This shows that an optimal solution cannot contain any batching scenario for furnace *j* with median coil *k*. □

LEMMA 3. If the solution of the LP relaxation problem of a B&B node is fractional, then there exists some coils *i* and *q* such that their θ_{iq} value is fractional.

PROOF: We prove it by contradiction. Suppose that θ_{iq} is integer for every possible pair of coils *i* and *q*. Given the fractional solution of the LP relaxation problem, then there must exist two batching scenario *s′* and *s″* both containing some coil *i* and both having a corresponding solution value $\lambda_j^{s'}$ or $\lambda_j^{s''}$ fractional. Since there are no duplicate columns in the optimal solution, there must exist a coil *q* which is not present in scenario s' but in s''. Thus $a_{qj}^{s'}$ = 0. According to the above analysis, we have the inequality that

$$\sum_{j=1}^{m}\sum_{s\in S_j} a_{ij}^s\lambda_j^s \geq \sum_{j=1}^{m}\sum_{s\in S_j} a_{qj}^s a_{ij}^s \lambda_j^s + \lambda_j^{s'}$$

By the definition of θ_{iq} and constraint (11), we have the following:

$$\theta_{iq} = \sum_{j=1}^{m}\sum_{s\in S_j} a_{qj}^s a_{ij}^s \lambda_j^s \leq \sum_{j=1}^{m}\sum_{s\in S_j} a_{ij}^s\lambda_j^s - \lambda_j^{s'} \leq 1 - \lambda_j^{s'}$$

Since $0<\lambda_j^{s'}<1$, we can conclude that $0<\theta_{iq}<1$, which contradicts with the assumption that θ_{iq} is integer for every possible pair of coils *i* and *q*. □

Appendix C: Detailed Description of the Tabu Search Heuristic

Below we first describe how an initial solution is generated (Section C.1), followed by a description of the three basic neighborhoods (Section C.2), a detailed procedure of the filter-and-fan method (Section C.3), and a step-by-step description of the tabu search heuristic (Section C.4).

C.1. Initial Solution

We propose the following greedy heuristic to obtain an initial solution for the tabu search heuristic. In each iteration, we first select an available furnace from the furnace type with the minimum number of remaining furnaces. Next we select a batch of coils to be loaded to the selected furnace as follows. All the available coils which can match with the furnace are sequenced in non-increasing order of their reward F_i. The coil with the largest reward value is chosen as the median coil of the batch. Then the other coils are checked sequentially, and the coils which can match with the median coil without violating the furnace height capacity constraint are put into the batch one by one. To guarantee a reasonable capacity utilization of the furnace, a weight threshold W (e.g. 70 tons, which is the lowest acceptable charging weight for a furnace) is applied. If the total weight of the batch is less than W after all the coils are checked, undo the batch by removing all the coils from the batch. Re-form the batch by choosing the coil next to the median coil chosen earlier in the coil sequence to be the new median coil of the batch and checking the coils and putting them into the batch sequentially as described earlier. This process is continued until a batch with a total weight of W or more is found. If no batch can be formed with a total weight of at least W when every coil in the coil sequence has been tried as the median, select the batch with the heaviest charging weight. The resulting batch is allocated to the furnace. This furnace and the coils allocated to it are then removed from further consideration. We repeat this procedure until there is no empty furnace. A complete initial solution is obtained.

C.2. Basic Neighborhoods

Based on the initial solution, a coil is either included in a batch for a furnace or not. We refer to the coils that are included in a batch in a solution as *inside coils* and the remaining coils as *outside coils*. To improve the solution using tabu search, certain moves within a neighborhood of the solution by exchanging some coils must be performed. We propose three neighborhoods in this section: one-to-one inside coil exchange neighborhood (N1), one-to-one inside-outside coil exchange neighborhood (N2) and one-to-two inside-outside coil exchange neighborhood (N3). These are explained in detail below.

N1: One-to-one inside coil exchange neighborhood. This neighborhood is defined as the set of solutions that can be obtained by an exchange between two coils which belong to different furnaces in the current solution. Moves in such a neighborhood can modify the assignment of the coils. When the coils in different furnaces are exchanged, the total charging weight will not be changed, but the matching quality may be improved.

N2: One-to-one inside-outside coil exchange neighborhood. This neighborhood is defined as the set of solutions that can be obtained by an exchange between an inside coil and an outside coil. Such a move can potentially increase the total charging weight and the matching quality as well.

N3: One-to-two inside-outside coil exchange neighborhood. For a furnace with a low charging weight, its residual capacity may not be enough to accommodate another coil. So a further neighborhood, one-to-two inside-outside coil exchange, is introduced here. A move in the neighborhood consists of an exchange between one inside coil and two outside coils. Such a move may increase the utilization of a furnace significantly.

In the tabu search algorithm, the search process goes through three phases repeatedly as follows to take advantage of different effects of the three neighborhoods. In the first phase, neighborhood N3 is used because of its potentially largest improvement on charging weight. Then the search is done with neighborhood N2 to improve the charging weight and the matching relationship. Finally, neighborhood N1 is adopted to improve the matching quality without changing the total charging weight. Because the residual capacities of some furnaces are changed in the last phase, it may increase the total charging weight by searching in neighborhoods N3 and N2 again.

C.3. Detailed Procedure of the Filter-and-Fan Method

We set the new solution as the root node S_0 of the search tree. The other nodes in the search tree are constructed as follows. Firstly, the basic neighborhood N1 is searched and the best η_1 solutions with the largest objective values are selected as nodes in the first level, which are denoted as ($S_1(1)$, $S_1(2)$,…, $S_1(\eta_1)$). For k = 1, 2, …, η_1, given the nodes generated in level k, the nodes in the next level k+1 are generated as follows. In level k, select η_1 best nodes with the largest objective values and denote them as $S_k(1)$, $S_k(2)$,…, $S_k(\eta_1)$. For each node $S_k(l)$, l = 1, …, η_1, select the furnace which has not been recorded in its parent node from the two furnaces where the coil exchange occurred when this node was generated. If both of the two furnaces where the coil exchange occurred are not recorded, select the furnace with a lower net reward. Record the selected furnaces as (j_1^k , j_2^k , …, $j_{\eta_1}^k$), each corresponding to one of the η_1 solutions. For each selected solution $S_k(l)$ in level k, search the basic neighborhood N1 between the coils in the recorded furnace j_l^k and the coils in the other furnaces, and then select the best η_2 solutions with the largest net rewards as the nodes in the next level k+1. This results in $\eta_1 \times \eta_2$ solutions in level k+1. The procedure terminates when a pre-specified maximum number of (e.g. 7) levels is reached or a node whose objective value is larger than that of the incumbent solution is obtained. Then select the node with the maximum net reward and remove all virtual coils and virtual furnaces from the solution corresponding to the selected node. Finally, set the newly generated solution to be the initial solution for next round of tabu search. Figure C-1 shows an example search tree.

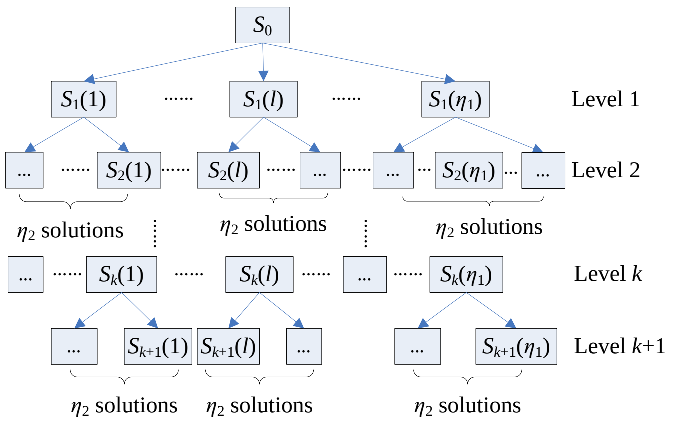


Figure C-1. Search tree for finding a compound exchange chain

In the search tree, a node in level k (k>1) is obtained by exchanging coils between the recorded furnaces of the node and its parent node in the solution corresponding to its parent node. Suppose that the node with maximum objective value is in the level K, then the corresponding solution can be obtained, from the solution of the root node S_0, by sequentially executing the exchanges of coils between the recorded furnaces. This series of exchanges of coils is called a *compound exchange chain*. Since the length of this chain is not pre-specified, the compound exchange chain is a variable depth neighborhood. An example of a compound exchange chain is given in Figure C-2. In the example, we only consider the reward for covering coils in the batching plan for illustration purposes. The reward and width of each coil in Figure C-2 are given in Table C-1. The furnaces drawn by dotted lines represent the virtual furnaces (furnaces j^2 and j^5), while the furnaces drawn by solid lines represent the real furnaces (furnace j, j^1, j^3 and j^4). Coils drawn by dotted lines are virtual coils. The total reward for the initial solution shown in Figure C-2 (before the compound exchange chain is performed) is 1122.68. The compound exchange chain can efficiently avoid the search procedure being trapped in a local optima.

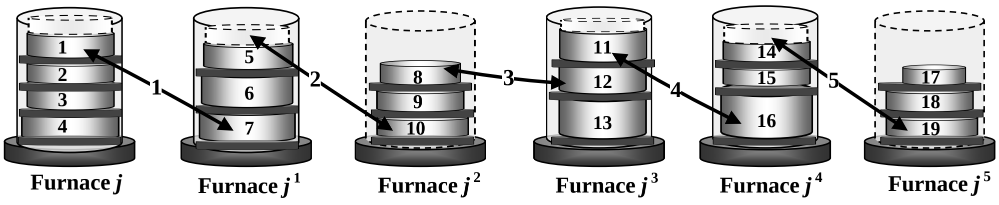


Figure C-2. An example of a compound exchange chain

Recall that the height of inner cover of furnace is 4700mm. Therefore, it is obvious that no improvement can be made by searching the neighborhoods N1, N2 and N3 on the initial solution in Figure C-2. However, after searching the variable depth neighborhood following the way shown in Figure C-2 by the filter-and-fan method, an improved solution is found as shown in Figure C-3, with the total reward improved to 1279.18.

Table C-1. The weight and width of each coil in Figure B-1

Number	Width	Reward	Number	Width	Reward
1	1015	82.4	11	1335	88.9
2	1015	82.79	12	1235	85.6
3	1015	82.79	13	1652	95.1
4	1015	83.1	14	1020	81.8
5	1015	81.5	15	1137	83.2
6	1312	88.5	16	1793	98.1
7	1265	88.9	17	985	74.6
8	1015	80.9	18	1020	79.7
9	1015	81.5	19	1012	80.1
10	990	81.1			

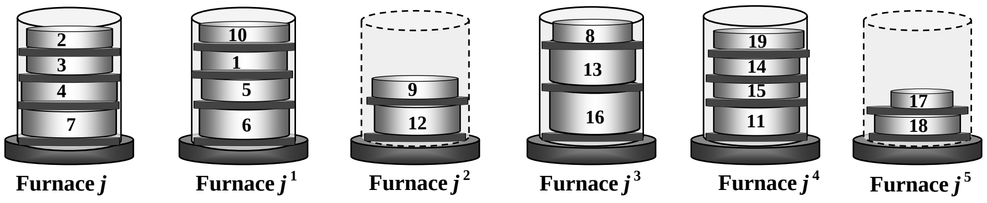


Figure C-3. The improved solution

C.4. The Complete Procedure of the Tabu Search Heuristic

Set iteration counter of the heuristic k = 0, and the corresponding upper limit T_{max} = 100. Let the number of iterations without improvement in the entire heuristic t = 0, and the corresponding upper limit T_{break} = 20. Denote the number of iterations without improvement in the searching process of each basic neighborhood t_1 = 0, and set the corresponding upper limit T_1 to be 5. The indicator *flag* is 1 if the incumbent solution has been updated, and 0 otherwise. The complete procedure of the proposed tabu search heuristic is presented as follows.

Step1 Get an initial solution σ_0 by the heuristic described in Section D.1. Let *flag*=0, the current solution $\sigma=\sigma_0$, and the incumbent solution $\sigma_{best}=\sigma_0$.

Step2 If $k > T_{max}$ or $t > T_{break}$, go to step 7; else if $t_1 > T_1$, let t_1=0 and go to Step 3; else, search neighborhood N3, and find the solution σ' in neighborhood N3 with the maximum objective value. If the objective value $f(\sigma') > f(\sigma_{best})$, then let t_1=0, *flag*=1, $\sigma=\sigma'$, $\sigma_{best}=\sigma'$;

else $t_1= t_1+1$, $\sigma=\sigma'$ Repeat Step 2.

Step3 If $t_1>T_1$, let $t_1=0$ and go to Step 4; else, find the solution σ' in neighborhood N2, that is not in the tabu list, with the maximum value of objective function. If the objective value $f(\sigma')>f(\sigma_{best})$, then let $t_1=0$, $flag=1$, $\sigma=\sigma'$, $\sigma_{best}=\sigma'$; else $t_1= t_1+1$, $\sigma=\sigma'$. Update tabu list, and repeat Step 3.

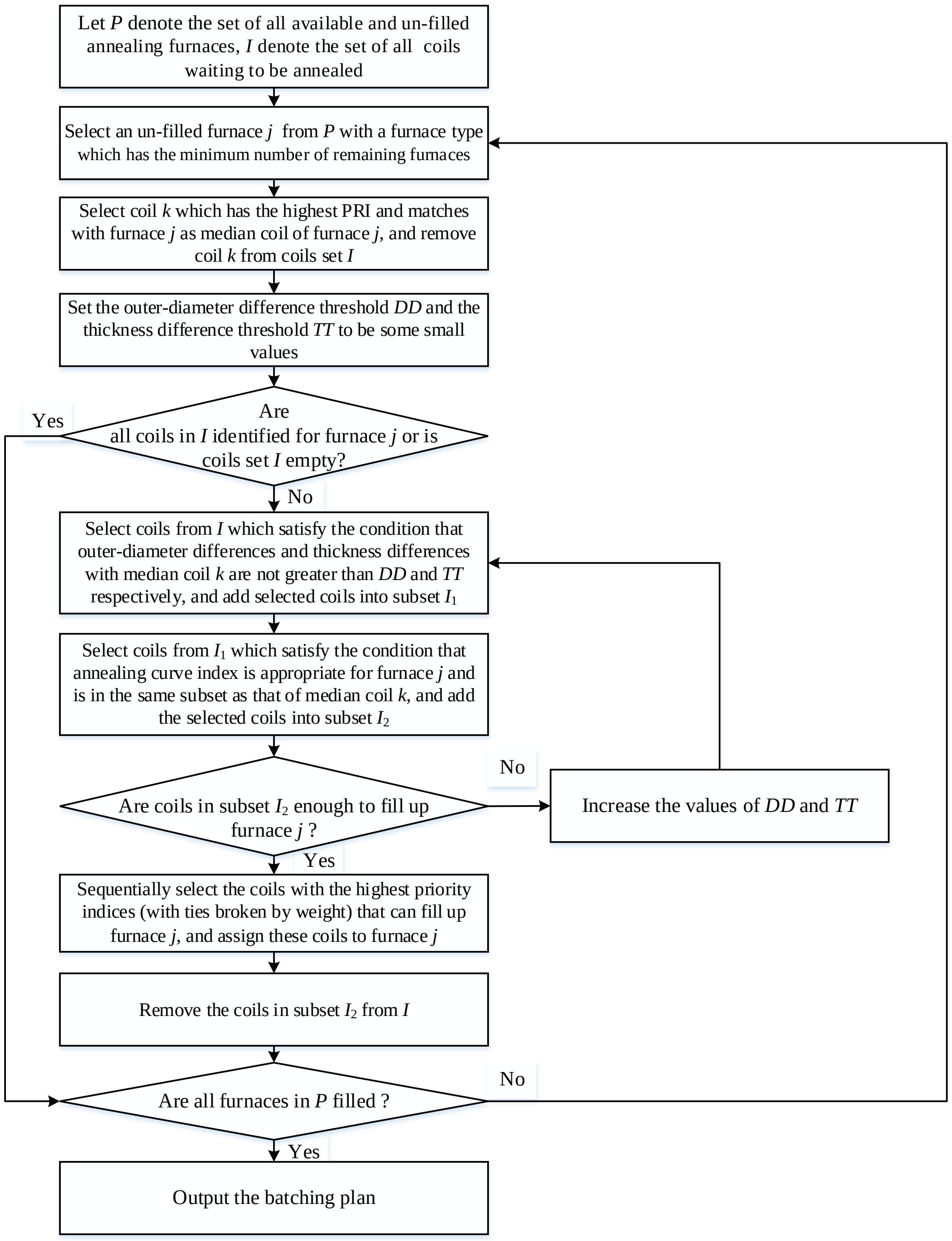
Step4 If $t_1>T_1$, go to Step 5; else, find a solution σ' in neighborhood N1, that is not in the tabu list, with the maximum value of objective function. If the objective value $f(\sigma')>f(\sigma_{best})$, then let $t_1=0$, $flag=1$, $\sigma=\sigma'$, $\sigma_{best}=\sigma$; else $t_1= t_1+1$, $\sigma=\sigma'$. Update tabu list, and repeat Step 4.

Step5 Let $k=k+1$. If $flag=1$, let $t=0$, $flag=0$, $t_1=0$, and go to Step 2.

Step6 Set the incumbent solution σ_{best} as the current solution. Perform filter-and-fan method in the variable depth neighborhood (described in Section 6.1 in the paper) to obtain a solution σ′ with the maximum value of objective function. If the objective value $f(\sigma')>f(\sigma_{best})$, $t=0$, $\sigma=\sigma'$, $\sigma_{best}=\sigma'$; else $t=t+1$, $\sigma=\sigma'$. Update tabu list, set $t_1=0$, $flag=0$, and go to Step2.

Step7 Output the solution σ_{best} and calculate the value of objective function for the final solution.

Appendix D: Flowchart of the Rule-Based Approach



Let P denote the set of all available and un-filled annealing furnaces, I denote the set of all coils waiting to be annealed
Select an un-filled furnace j from P with a furnace type which has the minimum number of remaining furnaces
Select coil k which has the highest PRI and matches with furnace j as median coil of furnace j, and remove coil k from coils set I
Set the outer-diameter difference threshold DD and the thickness difference threshold TT to be some small values
Are all coils in I identified for furnace j or is coils set I empty?
Yes
No
Select coils from I which satisfy the condition that outer-diameter differences and thickness differences with median coil k are not greater than DD and TT respectively, and add selected coils into subset I_1
Select coils from I_1 which satisfy the condition that annealing curve index is appropriate for furnace j and is in the same subset as that of median coil k, and add the selected coils into subset I_2
Are coils in subset I_2 enough to fill up furnace j ?
No
Increase the values of DD and TT
Yes
Sequentially select the coils with the highest priority indices (with ties broken by weight) that can fill up furnace j, and assign these coils to furnace j
Remove the coils in subset I_2 from I
Are all furnaces in P filled ?
No
Yes
Output the batching plan

Appendix E: Generation of Random Test Problems

The random test problems used in Section 7.2 are generated as follows. We follow the structure of the real data from Baosteel to generate four sets of random test problems with a wide range of sizes that are viewed as medium in practice. The first set of problems (denoted as S1) all have 40 coils and 4 furnaces; the second set (denoted as S2) 60 coils and 6 furnaces; the third set (denoted as S3) 80 coils and 8 furnaces; and the fourth set (denoted as S4) 100 coils and 10 furnaces. There are four types of furnaces involved in all the test problems. For each set of problems (with the given number of coils and furnaces), we generate 10 random problem instances as follows:

(1) The number of each type of furnace is randomly generated given the total number of furnaces.

(2) The heights of inner cover of all furnaces are all 4700mm.

(3) The outer-diameter of inner cover of small furnace is 2050mm, and that of big furnace is 2550mm.

(4) The widths of coils are generated uniformly from [800, 1800] mm.

(5) The thicknesses of coils are generated uniformly from [0.4, 3.8] mm.

(6) The weights of coils are generated uniformly from [10, 45] tons.

(7) The outer-diameters of coils are generated uniformly from [1600, 2500] mm.

(8) The annealing curve, steel grade, surface requirement, batching requirement, contract accumulation time, and storage time of coils are all selected from the practical data from Baosteel.

Appendix F: Description of the Decision Support System

The BDSS mainly consists of a data downloading module, a parameter setting module, an optimization module, an adjusting module, and an uploading module. The function diagram of the system is shown in Figure F-1. Some time before the start of every shift, the BDSS is run to make batching decisions for the upcoming shift using the latest furnace and coil information at the plant. The planners first run the data downloading module, which is connected to the Baosteel's central information system called 9672 system, to get the real-time data of the coils and empty furnaces in the annealing plant. Next, the optimization module is run to generate an optimal or near-optimal charging plan and show it on the screen. Both the exact branch-and-price-and-cut algorithm and the tabu search heuristic are imbedded in this module. Planners can choose one of these algorithms to run depending on the size of the problem they are facing. Planners can make adjustments on the plan through the adjusting module if they feel necessary. The final charging plan is uploaded to the 9672 system via the uploading module and transferred to the batch annealing plant to instruct the actual production. Figure F-2 is a screenshot showing a batching plan generated by the optimization module of the system.

Parameters used in the mathematical model and the algorithms need to be set to reflect the actual situations at the plant. A description of how the parameters F_i, C_{1ij} and C_{2ik} are set, which has been validated by planners in the batch annealing plant of Baosteel, is given in Appendix B. Planners can adjust the parameter values in the parameter setting module whenever they feel necessary.

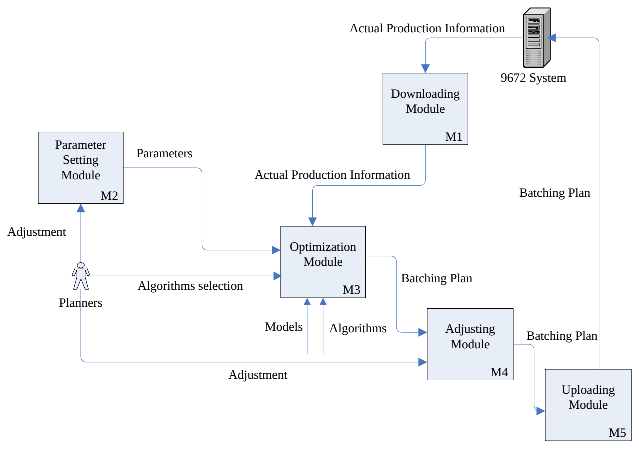
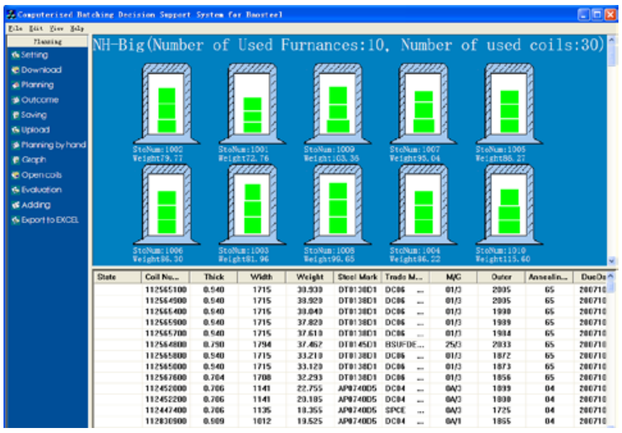


Figure F-1. A function diagram of the BDSS



NH-Big(Number of Used Furnances:10, Number of used coils:30)

StoNum:1002 Weight79.77
StoNum:1001 Weight72.76
StoNum:1009 Weight103.36
StoNum:1007 Weight95.04
StoNum:1005 Weight86.27
StoNum:1006 Weight86.30
StoNum:1003 Weight81.96
StoNum:1008 Weight99.65
StoNum:1004 Weight86.22
StoNum:1010 Weight115.60

State	Coil Nu...	Thick	Width	Weight	Steel Mark	Trade M...	M/C	Outer	Annealin...	DueD
	112565100	0.940	1715	38.930	DT0138D1	DC06 ...	01/3	2005	65	200710
	112564900	0.940	1715	38.920	DT0138D1	DC06 ...	01/3	2005	65	200710
	112565400	0.940	1715	38.040	DT0138D1	DC06 ...	01/3	1990	65	200710
	112565900	0.940	1715	37.820	DT0138D1	DC06 ...	01/3	1989	65	200710
	112565700	0.940	1715	37.610	DT0138D1	DC06 ...	01/3	1984	65	200710
	112564800	0.790	1794	37.462	DT0145D1	BSUFDE...	25/3	2033	65	200710
	112565800	0.940	1715	33.210	DT0138D1	DC06 ...	01/3	1872	65	200710
	112565000	0.940	1715	33.120	DT0138D1	DC06 ...	01/3	1873	65	200710
	112567600	0.704	1708	32.293	DT0138D1	DC06 ...	01/3	1856	65	200710
	112452000	0.706	1141	22.755	AP0740D5	DC04 ...	0A/3	1899	04	200710
	112452200	0.706	1141	20.185	AP0740D5	DC04 ...	0A/3	1800	04	200710
	112447400	0.706	1135	18.355	AP0740D5	SPCE ...	0A/3	1725	04	200710
	112830900	0.909	1012	19.525	AP0740D5	DC04 ...	0A/1	1865	04	200710

Figure F-2. A screenshot of a batching plan