Manufacturing & Service Operations Management

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Coil Batching to Improve Productivity and Energy Utilization in Steel Production

Lixin Tang, Ying Meng, Zhi-Long Chen, Jiyin Liu

To cite this article:
Lixin Tang, Ying Meng, Zhi-Long Chen, Jiyin Liu (2016) Coil Batching to Improve Productivity and Energy Utilization in Steel Production. Manufacturing & Service Operations Management 18(2):262-279. http://dx.doi.org/10.1287/msom.2015.0558

M&SOM MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 18, No. 2, Spring 2016, pp. 262–279
ISSN 1523-4614 (print) | ISSN 1526-5498 (online)

informs

http://dx.doi.org/10.1287/msom.2015.0558

Coil Batching to Improve Productivity and Energy Utilization in Steel Production

Lixin Tang, Ying Meng
Liaoning Key Laboratory of Manufacturing System and Logistics, Institute of Industrial Engineering and Logistics Optimization, Northeastern University, Shenyang 110819, China {lixintang@mail.neu.edu.cn, mengying@ise.neu.edu.cn}

Zhi-Long Chen
Robert H. Smith School of Business, University of Maryland, College Park, Maryland 20742, zchen@rhsmith.umd.edu

Jiyin Liu
School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, United Kingdom, j.y.liu@lboro.ac.uk

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, we show that our algorithms are superior to the 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 U.S. dollars and a large reduction of standard coal consumption and carbon dioxide emissions.

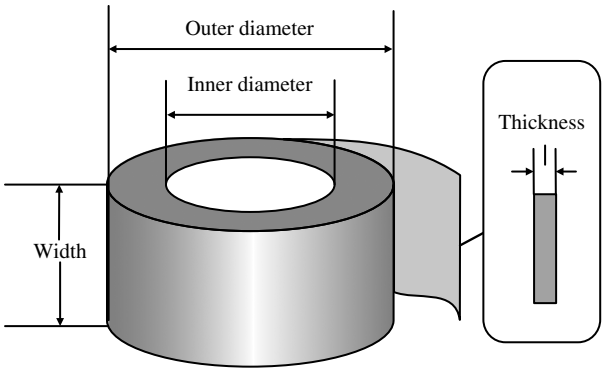
Keywords: steel production; batch annealing; batching decisions; integer programming; dynamic programming; branch-and-price-and-cut; tabu search
History: Received: July 4, 2014; accepted: July 5, 2015. Published online in *Articles in Advance* November 23, 2015.

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 as 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 among all steel products. Therefore, cold rolled products are a key to increasing profitability. The global annual output of cold rolled products accounts for approximately 40% of the total steel product output. The production process of cold rolled products consists of pickling, rolling, batch annealing, temper rolling, and some posttreatment steps. Batch annealing is a key operation that can improve the mechanical properties of cold rolled products. During the annealing operation, steel coils are first heated and soaked at a high temperature (i.e., 600–1,200°C) and then cooled to room temperature. This can change the structure of the crystal grain of steel coils and hence improve their thermodynamic stability and guarantee their performance. However,

Figure 1 The Structure of a Coil

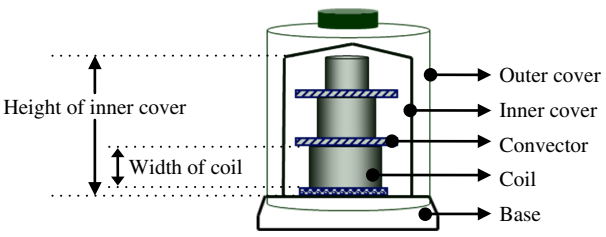


annealing is a main energy-consuming operation in the cold rolling stage; it consumes a huge amount of energy and resources, such as water, electricity, coal gas, and protective gas. Thus, the batch annealing process is one of the bottlenecks at most steel plants, and improving its efficiency becomes crucial to increasing 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 the 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.

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 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 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.

Figure 2 (Color online) The Structure of a Batch Annealing Furnace



In most steel plants, planners make coil batching decisions on a shift-by-shift basis such that the decisions in each eight-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 before 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 usually is approximately 1/10th 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 §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 the mismatching of coils in the same batch and the 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 before 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 multishift model will depend on the accuracy of the forecast of future coil arrivals, especially as a result of 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 before batch annealing. Hence, we choose to use the single-shift model as defined earlier, because 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 multishift 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 §7.1). Although planners may have extensive experience, it is very hard to construct effective batching plans manually because of 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 formally 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. Section 9 concludes the paper.

2. Related Literature and Problem Complexity

Batching problems arising in other industries of which we are aware, including the 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 the assignment of jobs to machines and the 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 are to be selected and processed into plates, which plates are 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 unidimensional 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 selecting ingots to be processed and deciding 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 of the batching problems reviewed above, 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, as 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 that 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 flow shop scheduling problem that arises from 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 that 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 a fully polynomial-time approximation scheme (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 it 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 as 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 because of the mismatching between the characteristics of these two coils.

Besides the matching requirements between coils described above, 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 that 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 Online Appendix A (online appendices available as supplemental material at http://dx.doi.org/10.1287/msom.2015.0558), 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 (IP) 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 that can be loaded into furnace j following the equipment and matching constraints described in §3.1 ($\bigcup_{j \in P} O_j = N$);
- R_k Set of coils that can be loaded together with coil k following the matching constraints described in §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 §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 §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 §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) max Z,

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

subject to

$$\sum_{j\in Q_i} X_{ij} \le 1 \quad \text{for } i \in N, \quad (2)$$

$$\sum_{i\in O_j} h_i X_{ij} \le H \quad \text{for } j \in P, \quad (3)$$

$$\sum_{k\in O_j} Y_{kkj} = 1 \quad \text{for } j \in P, \quad (4)$$

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

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

$$X_{ij} \in \{0,1\} \quad \text{for } j \in P,\ i \in O_j \quad (7)$$

$$Y_{ikj} \in \{0,1\} \quad \text{for } j \in P,\ k \in O_j,\ i \in O_j \cap R_k. \quad (8)$$

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 because of 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 that 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 that 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 §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* $[1], [2], \ldots, [q]$, *such that* $T_{[1]} \geq T_{[2]} \geq \cdots \geq T_{[q]}$, *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 Online Appendix B.

In the rest of this section, all the coils are reindexed such that $T_1 \geq T_2 \geq \cdots \geq T_n$, and $F_1 \geq F_2 \geq \cdots \geq F_n$. 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 nonoverlapping 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 nonoverlapping with that of all the coils assigned to any other furnace.*

Proof. See Online 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 that 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 $u = 1, \ldots, q$:

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

where

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

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

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 $f(j-1, x, q, u)$. In the latter case, coil j contributes $G(j, q, u-1)$ 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 $(j-1, x-1, y, y)$, for some y that maximizes $f(j-1, x-1, y, y)$.

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

PROOF. See Online Appendix B.

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

We first reformulate the integer programming formulation BDP given in §3.2 for the general problem as a set-packing model in §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 linear programming (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 §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 §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:

$$\text{(SP-BDP)} \quad \max Z,$$

$$Z = \sum_{j=1}^{m} \sum_{s \in S_j} c_j^s \lambda_j^s \tag{9}$$

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

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

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

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 0 (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 nonempty 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 that 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:

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

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 approximately one dozen arcs (i.e., there are approximately one dozen inequalities in (13)).

Because of 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 (Lübbecke 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 the 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 §5.2.1–5.2.3, 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 an 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 an 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), whereas in any integral solution this value must be at most 1.

Given a fractional solution of an 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:

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

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 time-tabling 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, $j = 1, \ldots, m$, and π_i, $i = 1, \ldots, n$, denote the dual variables corresponding to constraints (10) and (11), respectively; let β_l denote the dual variable value corresponding to the l-th valid inequality (14) (corresponding to Γ_l); and let 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 that can be loaded into a furnace of type j with the median coil k, following the equipment and matching constraints described in §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.

$$(\text{SUB}(j,k)) \quad \max\ \delta_{jk},$$

$$\delta_{jk} = \sum_{i \in N_{jk}} \Bigg(F_i - C1_{ij} - C2_{ik} - \pi_i - \sum_{v \in D_i} \gamma_{iv} + \sum_{u \in \{u \mid i \in D_u\}} \gamma_{ui} \Bigg) x_i - \sigma_j - \sum_{l=1}^{L} \beta_l z_l \tag{15}$$

subject to

$$\sum_{i \in N_{jk}} h_i x_i \le H - h_k, \tag{16}$$

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

$$x_i \in \{0, 1\} \quad \text{for } i \in N_{jk}, \tag{18}$$

$$z_l \in \{0, 1\} \quad \text{for } l = 1, \ldots, L, \tag{19}$$

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 Online 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 mixed integer programming (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 that 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):

$$\sum_{i \in N_{jk}} \Bigg(F_i - C1_{ij} - C2_{ik} - \pi_i - \sum_{v \in D_i} \gamma_{iv} + \sum_{u \in \{u \mid i \in D_u\}} \gamma_{ui} \Bigg) x_i - \sigma_j - \sum_{l=1}^{L} \beta_l z_l \ge \delta_{jk}^r. \tag{20}$$

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 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; $\boldsymbol{\delta}^*$ 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 Online 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 §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 $i, q \in N$,

$$V_i = \sum_{j=1}^{m} \sum_{s \in S_j} a^s_{ij} \lambda^s_j \quad \text{and} \quad \theta_{iq} = \sum_{j=1}^{m} \sum_{s \in S_j} a^s_{ij} a^s_{qj} \lambda^s_j.$$

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 Online 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 1 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 §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 §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 Online 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 exploring a variable-depth neighborhood is called the 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 $(S_1(1), S_1(2), \ldots, S_1(\eta_1))$. 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 $S_k(1), S_k(2), \ldots, S_k(\eta_1)$. For each selected node $S_k(l)$, $l = 1, \ldots, \eta_1$, 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 prespecified maximum number (e.g., 7) of 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 Online Appendix C.

7. Computational Experiments

As discussed in §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 size, and the problem with more than 100 coils in a shift is considered as large. 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-size 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 longer time 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 of a medium size.

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

Baosteel previously used a rule-based planning approach (described in §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 §§5 and 6 in the computational experiments. We collaborated with Baosteel in the last several years and developed a decision support system that replaced Baosteel's rule-based approach with our algorithms. The associated economic benefits for Baosteel are described in §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 §7.1. Computational results are reported in §§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 before our collaboration. A flowchart of the approach is given in Online 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 types 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 §3.1.3, a coil that has the highest PRI (with ties broken by choosing the coil with the largest weight) and that matches with the protective gas atmosphere of the furnace is selected as the median coil for the furnace. Select the coils that 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 from 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 reselect 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 §5, by comparing it with the MIP solver of CPLEX version 12.4 directly applied to the formulation BDP using randomly generated problem instances of a medium size. To test the effectiveness of the various special techniques developed in §5 (including the valid inequalities of §5.2.1, the variable reduction strategy of §5.2.3, and the valid inequality of §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 §5.2.1; (iii) BPCR, BP with both the valid inequalities of §5.2.1 and the variable reduction strategy of §5.2.3; and (iv) BPCRC, BPCR with the subproblems solved using the valid inequality developed in §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. Problems in the first set (denoted as S1) all have 40 coils and 4 furnaces; those in the second set (denoted as S2) have 60 coils and 6 furnaces; those in the third set (denoted as S3) have 80 coils and 8 furnaces; and those in the fourth set (denoted as S4) have 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 by following the structure of the real production data collected from Baosteel (see Online 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 (7,200 seconds) for each solution method. If an instance could not be solved to optimality within this time limit, we count its computation time as 7,200 seconds.

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 (of 10) whose integrality gap is 0 (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 (of the given 10) that are solved optimally within 7,200 seconds, respectively. Column "Time (s)" 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

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 (s)	Time red. r.t. CPLEX (%)	Solved to opt. by CPLEX
S1	BP	0.53	2	31.0	10	44.17	−6,034.72	10
	BPC	0.11	6	12.8	10	19.42	−2,597.22	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	56.0	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	466.0	8	2,247.96	37.71	5
	BPC	0.23	4	309.0	9	1,228.76	65.95	5
	BPCR	0.23	4	316.0	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	4,350.01	—	0
	BPC	0.07	4	219.5	8	1,812.52	—	0
	BPCR	0.07	4	277.1	9	1,019.16	—	0
	BPCRC	0.07	4	261.8	10	612.40	—	0

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 7,200 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 §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 and significantly reduces the integrality gap, hence reducing the B&B nodes that need to be explored and shortening 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 7,200 seconds. Our last conclusion is that for problems with 60 or more coils (S2–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 because of 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 §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 the 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 7,200 means that the problem is not solved to optimality within the time limit of 7,200 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

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	0.00	3.53	30.58
2	52	2	1	1	1	0.87	87.75	88.61	0.98	0.00	8.83	9.14
3	40	2	1	1	1	6.22	84.08	84.77	0.82	0.00	16.32	4.92
4	56	2	1	1	1	14.89	88.23	89.99	1.99	0.00	6.71	7.05
5	65	2	1	1	1	0.64	87.08	87.78	0.80	0.00	10.65	9.53
6	77	3	2	2	2	15.70	88.07	90.78	3.08	0.00	21.87	7,200.00
7	45	2	1	1	1	7.96	84.14	84.70	0.67	0.00	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	7,200.00
10	79	3	2	2	2	6.31	86.85	87.35	0.58	1.41	313.53	7,200.00
11	73	3	2	1	0	5.27	87.64	87.85	0.24	0.00	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	0.00	30.2	40.41
15	56	2	1	1	1	0.00	80.80	81.68	1.09	0.00	7.43	51.66
16	48	2	1	1	1	0.51	80.49	80.74	0.31	0.77	1,258.37	966.75
17	65	2	1	1	1	17.36	83.95	85.24	1.54	0.00	6.97	28.64
18	51	2	1	1	1	23.47	74.38	75.64	1.69	0.00	4.95	8.91
19	56	2	1	1	1	8.76	86.71	86.81	0.12	0.00	6.54	21.74
20	75	3	2	2	2	33.60	84.18	89.83	6.71	0.06	172.36	7,200.00
Average						8.83	85.43	86.55	1.32	0.14	103.88	1,502.27

to Baosteel by replacing their rule-based approach with our algorithms are estimated in §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 30–60 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 of a medium size in practice.

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 §7.1. The first version (denoted as BTABU) is the tabu search heuristic described in §6 using three basic neighborhoods but without using the variable-depth neighborhood. The second version (denoted as VTABU) is the improved tabu search heuristic that 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 20 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 the 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 times taken by BTABU and VTABU, respectively.

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 20 representative large 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 (s)" are the computation times (in seconds) used by the corresponding methods. Column "Gap (%)" is the relative gap between

Table 3 Comparing Two Versions of the Tabu Search Heuristic and the Branch-and-Price-and-Cut Algorithm

	Objective value		Average charging weight		Run time (s)	
Problem index	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	0.20	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	2.10	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	0.00	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

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 (UB-VTABU)/UB × 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 of the objective value represents improvement of the total reward of the selected coils and improvement of matching quality. Improvement of 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 §8.

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 §7.1) with very satisfactory performance. A description of the BDSS is given in Online Appendix F. Below we estimate the annual economic benefits of the BDSS to Baosteel by generalizing the test results given in §§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 four low-demand months and eight regular months.

By Baosteel's statistics, in the low-demand seasons in the last 4 years, an average of 2,160 batches of coils are annealed during the 4 low-demand months in a year. In the regular months of the same years, an average of 13,680 batches of coils can be annealed during the 8 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

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)		
		NH-big	NH-small	HH-big	HH-small	Imp1 (%)	Imp2 (%)	RULE	VTABU	Imp3 (%)	BTABU	VTABU	Gap (%)
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

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 25,538.4 tons ($= 1.12 \times 2{,}160 + 1.69 \times 13{,}680$). 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 approximately US$1.76 million ($= 68.93 \times 25{,}538.4$) 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 that 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 that can solve large instances to near optimality. Our computational experiments have demonstrated that our algorithms outperform the rule-based planning approach Baosteel previously used and that many other steel plants may still be using. We have collaborated with Baosteel and replaced their rule-based planning approach with a decision support system that contains our algorithms. The use of our algorithms has brought significant tangible and intangible benefits to Baosteel.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/msom.2015.0558.

Acknowledgments

The authors thank former editor-in-chief Steve Graves, the associate editor, and the two referees for their suggestions and comments, which have enabled the authors to significantly improve the paper. This research is supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China [Grant 71321001] and the Major International Joint Research Project of the National Natural Science Foundation of China [Grant 71520107004].

References

Aho AV, Garey MR, Ullman JD (1972) The transitive reduction of a directed graph. *SIAM J. Comput.* 1(2):131–137.

Avella P, Vasil'ev I (2005) A computational study of a cutting plane algorithm for university course timetabling. *J. Scheduling* 8(6):497–514.

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

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

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res*. 46(3):316–329.

Bianco L, Mingozzi A, Ricciardelli S (1994) A set partitioning approach to the multiple depot vehicle scheduling problem. *Optim. Methods Software* 3(1–3):163–194.

Chekuri C, Khanna S (2006) A PTAS for the multiple Knapsack problem. *SIAM J. Comput.* 35(3):713–728.

Chu CB, Antonio J (1999) Approximation algorithms to solve real-life multicriteria cutting stock problems. *Oper. Res*. 47(4): 495–508.

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

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

Glover F (1998) A template for scatter search and path relinking. Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D, eds. *Artificial Evolution*, Lecture Notes in Computer Science, Vol. 1363 (Springer, Heidelberg, Germany), 13–54.

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

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

Höhn W, König FG, Möhring RH (2011) Integrated sequencing and scheduling in coil coating. *Management Sci.* 57(4):647–666.

Kalagnanam JR, Dawande MW, Trumbo M, Lee HS (2000) The surplus inventory matching problem in the process industry. *Oper. Res*. 48(4):505–516.

Kellerer H, Pferschy U, Pisinger D (2004) *Knapsack Problems* (Springer, Berlin).

König FG, Lübbecke M, Möhring R, Schäfer G, Spenke I (2007) Solutions to real-world instances of PSPACE-complete stacking. Arge L, Hoffman M, Welzl E, eds. *Algorithms: ESA 2007*, Vol. 4698 (Springer, New York), 729–740.

Lee C-Y, Uzsoy R, Martin-Vega LA (1992) Efficient algorithms for scheduling semiconductor burn-in operations. *Oper. Res*. 40(4):764–775.

Lin J-H, Vitter JS (1992) ε-approximations with minimum packing constraint violation. *Proc. 24th Annual ACM Sympos. Theory Comput.* (ACM, New York), 771–782.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res*. 53(6):1007–1023.

Martello S, Toth P (1990) *Knapsack Problems: Algorithms and Computer Implementations* (John Wiley & Sons, Hoboken, NJ).

Megiddo N, Supowit J (1984) On the complexity of some common geometric location problems. *SIAM J. Comput*. 13(1): 182–196.

Möhring RH, Andreas SS, Stork F, Uetz M (2003) Solving project scheduling problems by minimum cut computations. *Management Sci.* 49(3):330–350.

Moon S, Hrymak AN (1999) Scheduling of the batch annealing process-deterministic case. *Comput. Chemical Engrg*. 23(9): 1193–1208.

Naphade KS, Wu SD, Storer RH, Doshi BJ (2001) Melt scheduling to trade off material waste and shipping performance. *Oper. Res*. 49(5):629–645.

Nemhauser GL, Wolsey LA (1988) *Integer and Combinatorial Optimization* (John Wiley & Sons, Hoboken, NJ).

Prasad P, Maravelias T (2008) Batch selection, assignment and sequencing in multi-stage multi-product processes. *Comput. Chemical Engrg*. 32(6):1106–1119.

Ranjbar M (2008) Solving the resource-constrained project scheduling problem using filter-and-fan approach. *Appl. Math. Comput*. 201(1–2):313–318.

Rego C, Duarte R (2009) A filter-and-fan approach to the job shop scheduling problem. *Eur. J. Oper. Res*. 194(3):650–662.

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

Tang LX, Jiang SJ (2009) The charge batching planning problem in steelmaking process using Lagrangian relaxation algorithm. *Indust. Engrg. Chem. Res*. 48(16):7780–7787.

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

Tang LX, Wang G, Chen Z-L (2014) Integrated charge batching and casting width selection at Baosteel. *Oper. Res.* 62(4):772–787.

Tang LX, Wang G, Liu JY (2011) A combination of Lagrangian relaxation and column generation for order batching in steelmaking and continuous-casting production. *Naval Res. Logist*. 58(4):370–388.

Tang LX, Xie X, Liu JY (2009) Scheduling of a single crane in batch annealing process. *Comput. Oper. Res*. 36(10):2853–2865.

Tang LX, Liu JY, Rong AY, Yang ZH (2000) A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron and Steel Complex. *Eur. J. Oper. Res.* 124(2):267–282.

Tang LX, Liu JY, Rong AY, Yang ZH (2001) A review of planning and scheduling systems and methods for integrated steel production. *Eur. J. Oper. Res*. 133(1):1–20.

Vonderembse MA, Haessler RW (1982) A mathematical programming approach to schedule master slab casters in the steel industry. *Management Sci*. 28(12):1450–1461.