Production design for plate products in the steel industry

S. Dash
J. Kalagnanam
C. Reddy
S. H. Song

We describe an optimization tool for a multistage production process for rectangular steel plates. The problem we solve yields a production design (or plan) for rectangular plate products in a steel plant, i.e., a detailed list of operational steps and intermediate products on the way to producing steel plates. We decompose this problem into subproblems that correspond to the production stages, where one subproblem requires the design of casts by sequencing slabs which, in turn, have to be designed from mother plates. The design of mother plates consists of a two-dimensional packing problem. We develop a solution approach which combines mathematical programming models with search techniques from artificial intelligence. The use of these tools provides two types of benefits: improvements in the productivity of the plant and an approach to making the key business performance indicators, such as available-to-promise at a production level, operational.

Introduction

Some items produced in the steel industry are steel coils (e.g., for automobiles and appliances), rectangular steel plates (e.g., for shipbuilding and construction), and steel blooms (e.g., for beams). In this paper, we describe an optimization model that captures many aspects of a multistage, rectangular plate production process in a major steel plant, and we describe a solution approach for this optimization model. Solving this optimization model yields a *production design* (PD), i.e., a detailed description of the production steps and related intermediate products, which yield a desired set of final plate products. We describe an implementation of our solution in an optimization tool intended for use at a plant operational level.

This PD tool was deployed at a large East Asian steel plant and is being used in daily production. Typical inputs consist of realized orders with processing start dates on each resource in a window of seven to ten days. The PD tool is intended to design and schedule the daily operations in the plant for a time horizon of one to two days. The main considerations in generating this design and schedule are the manufacturing constraints imposed by the machinery and a set of desired objectives that measure the productivity and efficiency of the operation design and schedule. The main challenge in developing this software system was to accurately model the very large number of manufacturing constraints and production-quality metrics, and to produce a feasible design and schedule within 30 minutes on a standard desktop computer.

Production process

We now describe the specific steel production process modeled in this paper. The goal of the production process is to satisfy orders for rectangular steel plates. *Orders* for plates (usually given by customers) specify the size of the rectangle, a thickness (usually between 10 mm and 100 mm), the steel grade, and the number of plates desired. The steel production process and a sample of its hierarchical elements are depicted in **Figure 1**.

Molten steel from the blast furnace is *refined* in the basic oxygen furnace (BOF) in batches of about 250 to 300 tons, each batch resulting in steel with different metallurgical properties. These batches of refined molten steel are poured through a mold (or simultaneously through two molds) with a rectangular cross section; this process is called *casting* and is performed in a *continuous caster*. The strand of metal emerging from a mold is cut across its length to form a *cast slab*, which is again cut

0018-8646/07/$5.00 © 2007 IBM

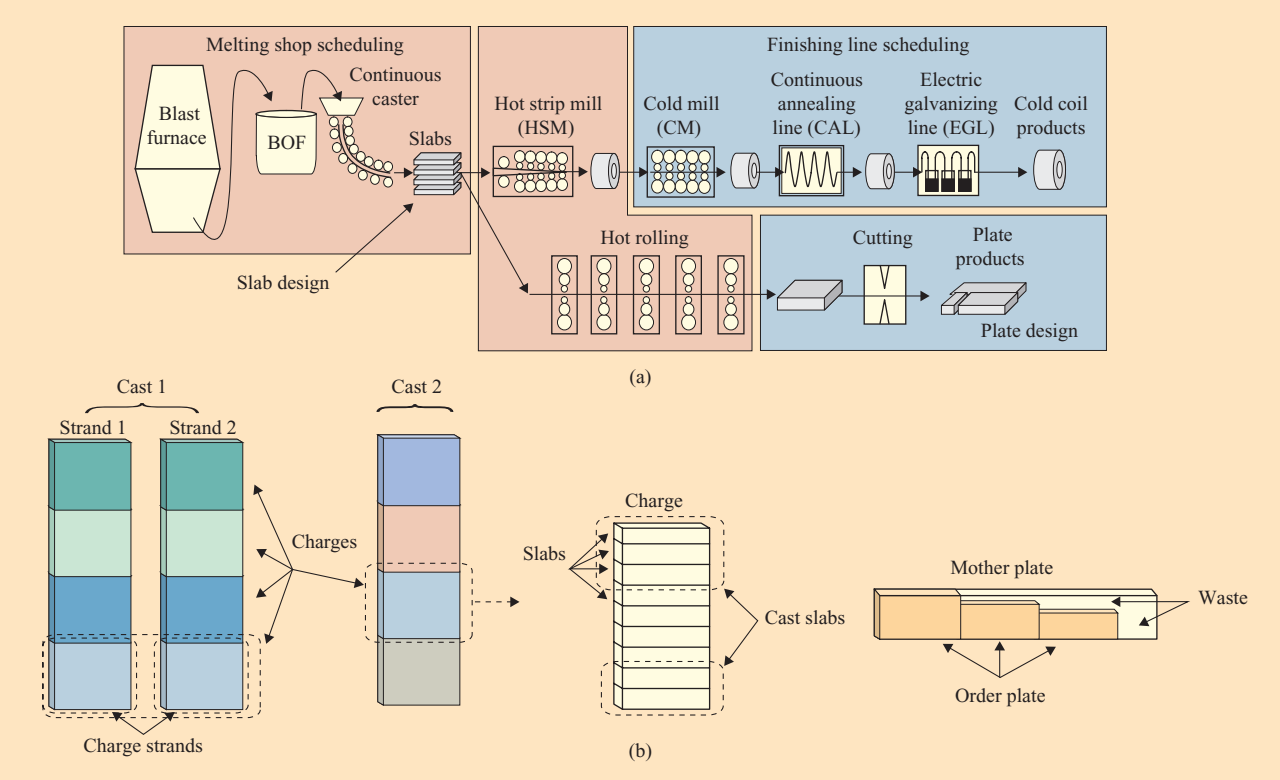

Figure 1

Steel plate production: (a) Production process; (b) example of the hierarchical relationship of the elements.

across its length to form a *slab* of steel. Slabs and cast slabs are cuboidal objects with the same width and thickness as the mold used in making them. Slabs have a width between one and two meters and a length between two and five meters; a cast slab contains two or three slabs. The slab thickness, and thus the mold thickness, lies in a small set of allowable thicknesses in the range 200 mm to 400 mm.

The slabs are processed further, for instance by quenching or heat treatment, to impart different properties to them; these properties are encoded as steel grades. We refer to a sequence of slabs created from the same mold (or pair of molds if two molds are used simultaneously) as a *cast*, and a sequence of slabs from a single mold as a *strand*. Thus, a cast can consist of one or more (usually two) strands. If a single mold is used, a cast and a strand are identical notions. A cast and the strands in the cast satisfy certain geometry restrictions: All slabs in a strand have the same width and thickness, two strands in a cast have the same thickness, and the length difference between them (the length of a strand is the sum of the lengths of slabs in the strand) is restricted. We refer to a batch of steel and the slabs produced from it as a *charge* and the part of a strand associated with a charge as a *charge strand*. A cast can thus be viewed as a sequence of charges, and a strand as a sequence of charge strands. As slabs in a charge have the same chemical properties and are only modified mechanically later on, the steel grades of slabs in a charge can take on only limited sets of values. In addition, slabs in consecutive charges are required to have similar grades. We call these restrictions *charge-grade* constraints and *grade-transition* constraints, respectively, and describe them in the section on problem specification below. There are a number of additional constraints for casting, which we describe later.

A slab is subsequently *rolled* (thickness decreases and area increases) into a *mother plate* with the same volume and steel grade as the slab and a rectangular size of up to 5 m × 50 m. Mother plates are finally cut across their width and length into *order plates*—plates corresponding to specific orders—with the same thickness and steel grade. Thus, a mother plate can be viewed as a two-dimensional pattern of order plates, though in our application more than 95% of mother plates have order plates arranged in one-dimensional patterns. Throughout

this paper, we assume the previous simplified model of the actual steel grade constraints.

The rolling machines have restrictions on the dimension transformations they can perform in creating mother plates from slabs, and slabs and mother plates have minimum and maximum length and width restrictions, discussed in the next section. Some slabs, called *surplus slabs*, are produced without a target mother plate in mind, but simply to satisfy the *charge-weight constraint*, i.e., the condition that a charge consists of, say, 250 to 300 tons of slabs. We call the other slabs *order slabs*, because order plates are produced from them. The different facilities and machines associated with the above production steps—such as the casters, the rolling mills on which mother plates are rolled, and the refining facilities—have daily capacity limits, and, for some of the facilities, minimum usage targets. The usage targets are imposed for productivity reasons. There are some additional manufacturing steps and constraints involved in plate production; some we did not model, and some we do not describe because of lack of space.

In Figure 1(b), we show the hierarchical relationships among the different intermediate and final products of the steel production process. We depict a cast with two strands in cast 1 and one with a single strand in cast 2. Each charge consists of one or two charge strands, each charge strand being a sequence of cast slabs, where each cast slab consists of two to three slabs. Each order slab corresponds to a mother plate; the mother plate, shown at the right side of Figure 1(b), consists of three order plates that correspond to orders with the same thickness but different widths. The differing widths, and the fact that mother plates have minimum length requirements, result in some waste, which is depicted by the light-colored area on the mother plate. We define the *PD problem* for plates as the problem of determining the following:

1. The number of order plates that will be manufactured for each order.
2. The set of mother plates from which the order plates will be cut, along with the position of the order plates on the mother plates.
3. The set of slabs from which the mother plates will be rolled, one per mother plate.
4. The set of charges along with the constituent cast slabs and slabs in each charge, and the location of the slabs in the charge strands.
5. The set of casts, and for each cast, its constituent charges and their sequence in the cast.

These must be determined while satisfying all capacity limits, targets, and manufacturing constraints, and maximizing various objectives or production metrics. For example, one objective is to maximize the number of orders fulfilled by their due dates, another is to minimize waste, and another is to maximize the average number of charges per cast. We describe the objectives and metrics in later sections. We refer to the above items 1 and 2 together as the *mother-plate-design* (MPD) *problem*; 1, 2, and 3 together as the *slab-design* (SD) *problem*; and 4 and 5 together as the *cast-design* (CD) *problem*.

An additional manufacturing process we model is the following. The plant usually has some plates, slabs, and cast slabs in inventory. We define the *inventory-allocation problem* as the problem of deciding which order plates to manufacture from inventory, and assigning these to specific inventory items (slabs are converted to order plates by rolling and cutting).

In later sections, we give more detail while attempting to abstract the most interesting constraints, objectives, and computational issues in our application. We emphasize that we describe a real-life model and implementation in this paper and not just an abstract model, and we deal with too many constraints and objectives to be able to list them fully here.

Related work

We are not aware of any optimization model in the literature that deals with the PD problem for plate products in its full generality. In a survey on the use of mathematical programming applications in the steel industry, Dutta and Fourer [1] suggest that "cutting stock optimization to maximize overall yield of multistage production processes" in steel plants was not addressed prior to their survey, and such work "would go beyond most previous work on the cutting stock problem, which has used single stage models." In recent, related work, Moreno et al.[1] study a PD problem for steel billets and use multiple mixed-integer programming models to solve it. The problem they study can be viewed as a special case of the PD problem defined in this paper. In their problem, there is no distinction between slabs and order plates, because the rolling and cutting steps are not performed on slabs. Second, a cast slab contains slabs associated with a single order, and finally, they generate a single cast with a predetermined width. See also Menezes et al. [2]. In other work, Harjunkoski and Grossman [3] and Chang, Chang, and Hong [4] study variants of the PD problem for coil products made using continuous casters. See Pacciarelli and Pranzo [5] and Lee et al. [6] for other production scheduling problems in the steelmaking industry, and for solution techniques for these problems.

Some of the other subproblems defined above clearly resemble well-known optimization problems. The MPD

[1]L. Moreno, M. Poggi de Aragão, O. Porto, and E. Uchoa, "A MIP Approach to the Continuous Casting Production Planning," manuscript available from the authors.

problem is closely related to (though more general than) the two-dimensional cutting stock problem (2D CSP). We emphasize that mother plates do not resemble stock in that they do not have prescribed dimensions, and their dimensions have to be determined. In spite of this difference, one can use the standard solution approach for the 2D CSP, namely delayed column generation, where columns represent 2D patterns of order plates. See Gilmore and Gomory [7, 8] and Vanderbeck [9] for work on the 2D CSP. Further, it is possible to relate the MPD problem to the multiple-class integer knapsack problem with setups described in Perrot and Vanderbeck [10]. In that problem, the items belong to different classes, and if a class is used, the weights of items chosen for a class lie in a weight range. These constraints are similar to, though a special case of, the grouping constraints described in the next paragraph. Vonderembse and Haessler [11] describe a problem related to the division of cast slabs ("master slabs" in their terminology) into slabs, and a solution approach implemented at Bethlehem Steel.

The PD problem is fairly complex, as the MPD subproblem already generalizes the 2D CSP. Further, there is a nonlinear relationship between slab dimensions and the corresponding mother-plate dimensions. Modeling the entire problem as a single mixed-integer linear program, which is practically solvable by exact branch-and-bound methods, is not a realistic option. We tackle the PD problem by decomposing it into an SD problem and a CD problem. Our goal in the SD problem is to create a collection of mother plates or 2D patterns of orders whose properties meet the following restraints:

- *Dimension constraint*—The mother plates can be rolled from slabs with a common thickness and width.
- *Grouping* (*by grade*) *constraints*—The collection can be partitioned into subcollections, with two conditions:
  - *Condition 1*—Each subcollection of slabs satisfies the charge-grade constraints.
  - *Condition 2*—The weight of a subcollection allows it to be further partitioned into an integral number of charges.

Condition 2 is equivalent to saying that there is an integer $t > 0$ such that the weight of a subcollection lies in the range 250 t to 300 t. The MPD problem can be viewed as the SD problem without the dimension constraint. In the Solution overview section below, we explain how we fix the thickness and width of slabs in the SD problem, thereby eliminating the dimension constraint. Finally, our goal in the CD problem is to use slabs generated in the SD problem and generate a set of feasible casts. We emphasize different objectives in the SD and CD problems, though minimizing waste and maximizing the number of orders fulfilled by their due dates are important in every phase of our solution approach. We essentially solve the SD and CD problems by means of column generation; in CD, the columns correspond to casts, and in SD, the columns correspond to mother-plate patterns and slabs. To generate a single cast, we solve an SD problem. We describe our solution approach in more detail below.

The PD optimization tool in which we implemented our solution approach consists of different modules corresponding to the different subproblems defined above. The typical flow of activities in planning daily production using the PD tool is as follows. An updated order book (list of orders) is provided as input. The order book contains a set of realized or planned orders with associated due dates—usually based on some available-to-promise (ATP) analysis—and a processing date on each machine or resource based on a rough capacity plan. The first step is to match the order book to available inventory using the inventory allocation (IA) module. Typically less than 10% of the order book is handled from inventory. To design the order book for manufacturing, the PD problem is solved by iteratively invoking the SD and CD modules.

The remainder of this paper is organized as follows. We first give a detailed description of the PD problem, and then an overview of our solution approach for this problem emphasizing the interactions between the SD and CD modules. We then discuss the SD module and the IA module, followed by a description of the CD module. We discuss some computational issues, provide a summary of the efficiency gains that can be achieved using the PD tool, and report results.

Problem specification

We now set out in more detail the specific problem we solve in this paper in terms of inputs, outputs, and constraints for production design. The inputs to the problem are the following:

- An order book, i.e., a set O of orders.
- A set C of casters, and for each caster $i \in C$, a set T_i of mold thicknesses.
- A set R of rolling mills.
- A set $G = \{g_i\}$ of grades.
- A set H of mixable grade sets, where each grade set $S_i \in H$ is a subset of G.
- A directed graph $GT = (V, E)$ representing allowed grade transitions between consecutive charges; each node in V corresponds to a grade set in H, and an arc $(i, j) \in E$ from node i to node j implies that a charge with grade set i can be followed by a charge with grade set j in a cast.

- A set F of facilities (including the casters), and for each facility $f \in F$, an upper bound on its capacity U_f and a minimum usage target L_f. The upper bounds give the maximum amount of steel that can be processed on these facilities. The usage targets are usually zero except for the rolling mills. For individual casters and refinement facilities within casters, the bounds are given in numbers of charges that can be processed.

Each order in the order book comes with a minimum and maximum number of plates. If the number of designed plates for an order lies within the specified minimum and maximum values, the order is said to be complete. Orders with due dates within a fixed number of days of the current date (usually three) are designated as rush orders. Each order has a grade from the set G, and orders in a mother plate have the same grade. A mother plate has the same grade as its order plates, and so does the slab from which it is rolled. Orders in a mother plate can have different widths and lengths and can be packed in one-dimensional (simple) patterns, shown in **Figures 2(a)–(c)**, or two-dimensional (mosaic) patterns, shown in **Figure 2(d)**. The parameters $L_{\min}$ and $L_{\max}$ specify the lower and upper bounds on the lengths of the mother plates. These are not constant and depend on the thickness, width, target slab geometry, and planned route of the mother plate, but not on its one- or two-dimensional nature. For simple patterns, the number of order plates, the number of distinct orders, and the difference in width between the widest and narrowest orders are all bounded above by specified numbers. Orders with different widths, as in Figure 2(b), result in (vertical) waste. If the combined length of the orders is less than $L_{\min}$, as in Figure 2(c), the region to the right of the orders is treated as (horizontal) waste. One can add a surplus plate, as in Figure 2(d), such that the length of the surplus plate plus the combined order length is $L_{\min}$ or more, thus avoiding horizontal waste. A surplus plate does not correspond to any current order, but to a potential future order, and has a minimum and maximum length derived from the expected future order. Surplus plates also have a minimum and maximum length, which comes from the expected minimum and maximum lengths of future orders. We call a set of orders that can be placed together on the same mother plate an *order component*.

Each caster $i \in C$ makes casts with at least l_i and at most u_i charges, with each charge weighing between wl_i and wu_i tons. The thickness of a cast made in caster i must lie in the set T_i, and all slabs in a cast must have the same thickness. A cast can have one or two strands; we discuss only double-strand casts in this paper. In such casts, all slabs in a strand have the same width. The widths of the two strands can be different, but here we discuss only

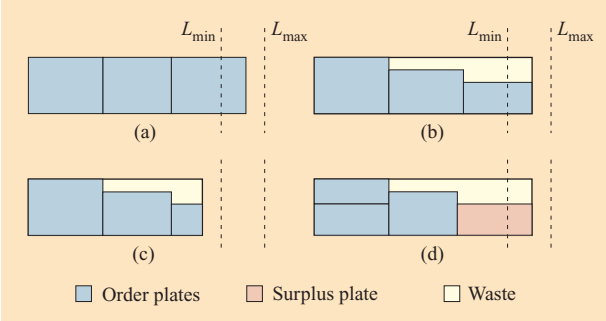

Figure 2

Possible arrangements of orders and surplus plates on a mother plate.

equal-width strands. Each charge in a double-strand cast has two strands, and the width and thickness of a charge strand are the same across all charges in the cast. The set of slab grades in a charge must lie in H. Two charges can be adjacent in a cast if and only if they have compatible grade sets as given by the graph GT (i.e., there must be an arc between their grade sets in GT).

For a slab, its *route* stands for the combination of the caster on which it is manufactured, the mold thickness used, and the rolling mill where it is converted to a mother plate. A mother plate has the same route as the slab from which it is made. Each rolling mill has restrictions on the minimum and maximum slab lengths and widths it can handle. These restrictions vary with the slab thickness. For a given slab thickness, a rolling mill can handle any slab with lengths and widths between the corresponding minimum and maximum widths and lengths. In other words, for a given thickness, the minimum and maximum slab lengths are independent of its width. A slab and its corresponding mother plate have the same mass and volume (because steel density does not change). A mother plate has a width and length between a minimum and a maximum value. These values are functions of the dimensions and route of the slab from which it is rolled and the thickness of the mother plate. Further, the minimum and maximum lengths of a mother plate are functions of its width. We give additional constraints on casts, charges, slabs, and mother plates when we discuss our solution modules.

Some important metrics are the following:

- *Yield ratio*—The ratio of the total weight of order plates and surplus plates to the total weight of mother plates.
- *Surplus ratio*—The ratio of the total weight of surplus plates to the total weight of mother plates.

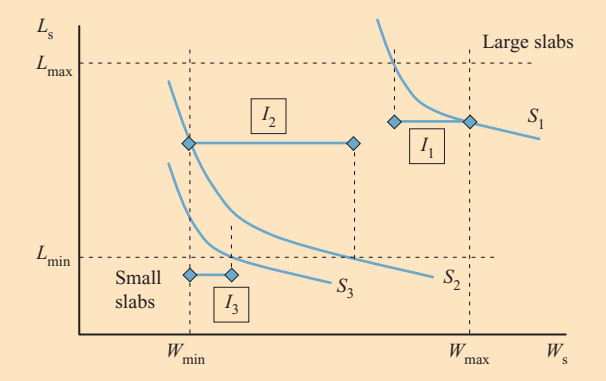

Figure 3

Slab width–length curves for mother plates with fixed volume.

- *Average slab (or mother plate) weight*—The total weight of slabs (mother plates) divided by the number of slabs (mother plates) manufactured.
- *Surplus-slab ratio*—Given by the total weight of surplus slabs divided by the total weight of slabs.
- *Rush-completion ratio*—The ratio of completed rush orders to the total number of rush orders.

The most important objectives for the PD problem are maximizing the rush-completion ratio, the yield ratio, and the average slab weight while minimizing the surplus ratio and the surplus-slab ratio.

In our application, the inputs to the PD problem consist of about 3,000 to 5,000 orders, each with a demand for from one to 100 plates, with most orders being for only one to five plates. There are about three or four casters and about 15 to 20 different caster and mold thickness combinations; i.e., the set $T = \cup_{i \in C} T_i$ has about 15 elements. There are about three or four rolling mills. The set G consists of 50 grades or so, and the set H has up to 150 subsets of G. The solution of the PD problem usually consists of about 1,000 mother plates of ten tons weight on the average, arranged in five to ten casts, each consisting of five to eight charges per cast, for a total of about 10,000 to 15,000 tons of steel in a solution (equivalent to a day's production).

Solution overview

In this section, we provide an overview[2] of the solution approach and the flow of the engine in terms of how the modules are invoked. Within this context, we provide a description of how the geometry transformations among a slab, a mother plate, and the geometry constraints at the casts form a central thread of interaction between the different modules.

Recall that the volume of a slab equals that of the mother plate from which it is rolled. Let t_p, w_p, l_p stand for the thickness, width, and length of a mother plate, respectively, and let t_s, w_s, l_s stand for the corresponding dimensions of the slab from which it is rolled. Clearly $t_p \times w_p \times l_p = t_s \times w_s \times l_s$. For a given slab thickness $\bar{t}$, let $[W_{min}, W_{max}]$ be the allowed slab width range, and let $[L_{min}, L_{max}]$ be the allowed slab length range. Suppose we fix the desired slab thickness to $\bar{t}_s$. Then a slab can have a volume in the range $[\bar{t}_s W_{min} L_{min}, \bar{t}_s W_{max} L_{max}]$. Now consider the possible mother-plate dimensions from slabs with the above range of volumes and assume that we fix the mother-plate thickness and width to that of some order plate. Then a feasible mother plate has an allowed length range contained in $[k_{min}, k_{max}]$, where k_{min} and k_{max} equal, respectively, the minimum and maximum slab volume divided by the product of the mother-plate thickness and width. Conversely, if we take a specific mother plate with volume V, a slab with thickness $\bar{t}_s$ can be rolled to give the above mother plate if $w_s \times l_s = V/t_s$, w_s is contained in $[W_{min}, W_{max}]$, and l_s is contained in the range $[L_{min}, L_{max}]$. In other words, the minimum allowed width is the larger of $V/(t_s \times L_{max})$ and W_{min}, and the maximum allowed width is the smaller of $V/(t_s \times L_{min})$ and W_{max}. We depict this relationship between the slab and mother-plate geometries in **Figure 3**. The x-axis represents slab width W_s and the y-axis represents slab length L_s. Each curve represents a mother plate of constant weight and the possible geometries of the associated slab; i.e., a slab with width and length given by a point on a curve can be rolled into the corresponding mother plate. Clearly S_1 has the largest volume and S_3 the smallest volume. We depict W_{min} and W_{max} by vertical lines and L_{min} and L_{max} by horizontal lines. The following are a few useful observations from this graph:

- The width range for each slab can be represented by an interval (intervals I_1, I_2, and I_3 stand for the allowed width ranges for the three slabs).
- Large (heavy) mother plates (in the upper right corner) and small (light) mother plates (at the bottom left corner) have little flexibility in their slab geometries.
- The width range of a slab can be increased by decreasing (for large mother plates) or increasing (for small mother plates) the size of the mother plates.

Suppose we fix the desired slab thickness to $\bar{t}_s$ and the desired slab width to $\bar{w}_s$. The allowed slab length range remains $[L_{min}, L_{max}]$. Therefore, a slab can have a volume in the range $[\bar{t}_s \bar{w}_s L_{min}, \bar{t}_s \bar{w}_s L_{max}]$, and a mother plate with

[2]Based on S. Dash, J. Kalagnanam, and C. Reddy, "Method for Production Design and Operations Scheduling for Plate Design in the Steel Industry," U.S. Patent Application No. 20060100727, May 2006.

thickness and width fixed, as in the previous paragraph, has an allowed length range $[l_{min}, l_{max}]$, where l_{min} is at least k_{min} and l_{max} is at most k_{max}. Certain values of l_{min} and l_{max} are not desirable in the sense that they may lead to a lot of surplus plates or waste on the mother plates. For example, if all order plates have a length of 10 m, surplus plates are required to be at least 4 m in length, and l_{min} and l_{max} are 12 m and 13 m respectively, then the only feasible mother plates either have only surplus plates or at least 2 m of waste. It is therefore clear that slab width is a crucial parameter in our problem, and identifying *good* slab widths (widths at which the surplus or waste is minimal) is a central issue.

This suggests one possible mixed-integer programming (MIP) approach to modeling the nonlinear relationship between the slab and mother-plate dimensions: Enumerate every possible slab thickness and width combination. There are only about 15 possible values of slab thickness, but about a thousand possible width values, since slabs can have an integral width between 1 m and 2 m, and therefore about 15,000 combinations. One can then design a column-generation approach for the CD problem based on this idea. Let the columns stand for casts of different thicknesses and widths, and in the pricing subproblem (finding casts with negative reduced costs), solve an SD subproblem for each of the 15,000 combinations. This is clearly a wildly impractical approach because, given the complexity of the SD subproblem, we are able to approximately solve only a few hundred SD subproblems in 30 minutes—the time limit in our application.

Our approach is influenced by the above impractical column-generation approach in that we work with a formulation in which the columns are casts that overlap, in the sense that they use some of the same orders. However, we attempt to identify a small set of good widths quickly, and we now describe how we do this.

Assume that we have fixed a caster and a slab thickness for the caster, but no prescribed slab width. As discussed earlier, this imposes a slab and mother-plate volume range. We then solve the MPD problem (described in the Introduction) to get a collection of mother plates, which is, in a sense, the best possible collection of mother plates if the grouping constraints are considered and the dimension constraint ignored. We then compute the width interval for the slab (as in Figure 3) from which the mother plate can be rolled. We analyze these width intervals to obtain good widths. For example, if the MPD solution consists only of the mother plates in the figure, any width in the interval I_3 is good in the sense that two mother plates in the solution can be rolled from slabs with that width, whereas widths greater than the maximum width in I_2 are not so good, because either only one or no mother plates can be rolled from slabs of those widths.

We represent each slab with a node in a graph and introduce an arc if slabs have an overlapping width interval to obtain an *interval graph*. We can now easily determine slab clusters that have a common width among them by enumerating the maximal cliques[3] in this graph. As the CD problem focuses on generating a collection of slabs with a common width, the interval graph representation is very useful for identifying candidate clusters of slabs to compose casts. For each slab cluster corresponding to a maximal clique, we analyze the number of good charges (i.e., charges where surplus slabs do not have to be added to satisfy the charge weight constraint) that can be formed from the slabs, and the number of casts that can be formed from these charges. This analysis yields a collection of good widths (more precisely, width intervals). We call this process *width exploration*. We then select a small set of widths from these intervals, and for each of them, fix the slab width and solve the MPD problem. We use these mother plates to create candidate charges and casts, and thereby columns of our CD problem MIP formulation (which is a set-packing formulation). We iteratively select a collection of casts, remove the orders used in these casts, generate a few more casts, and select more casts until we use up the casting capacity. The question naturally arises whether the quality of the casts that are selected can be improved by making use of the remaining casts (unused slabs in these casts). Note that each invocation of the MPD problem so far has taken into consideration only the specific width for which we were generating a candidate cast, and not the entire collection of widths.

We take the selected casts and treat them as templates; i.e., we extract from each cast its thickness and width, and the grade sets of the different charges, but throw away the specific slabs in them. The quality of selected casts can be improved (often substantially) by redesigning the mother plates from scratch to the geometry and grade specified by the templates. These geometry requirements for the slabs can be translated into length ranges for the mother plates to be designed. Observe that we can now solve an MPD problem without the grouping constraints, as we have already decided on the grade sets of the charges. We call this process *template filling*.

Slab design

Mother-plate design

In the MPD problem, we take as input a list of orders and a target slab geometry (specified by a caster, a

[3] A *clique* is a set of nodes in a graph such that there exists an edge between any two nodes in the set. Another way of defining this is that each node shares an edge with every other node in the set.

slab thickness, and a slab width range) and output a list of mother plates along with the locations of order plates on each mother plate. Each output mother plate can be rolled from slabs with the target geometry. We repeatedly solve the MPD problem with different inputs and different parameters. We first describe the general problem we solve, and then the specific variations in different invocations of mother-plate design. We model this problem as a 2D CSP with capacity constraints and the grouping constraints described above in the section on related work.

Each order has an associated list of facilities at which it requires chemical or mechanical processing. The capacity constraints reflect the fact that the total weight of order plates processed at a given facility cannot exceed a given capacity for the facility.

In our application, we considered the following objectives. Given a set of designed mother plates, the order-completion ratio equals the number of completed orders divided by the number of used orders. An order is *used* if at least one plate for the order is designed. The five requirements were the following:

a. Maximize number of completed rush orders.
b. Maximize order-completion ratio.
c. Maximize yield ratio.
d. Maximize average mother-plate weight.
e. Minimize surplus ratio.

Obviously one cannot handle all five of these requirements simultaneously. Let y_c (y_u) be vector-valued variables such that the ith component of y_c (y_u) is 1 if the ith order is complete (used), and 0 otherwise. We can model the first objective (let O_R stand for the set of rush orders) as

$$\max \sum_{i \in O_R} (y_c)_i .$$

It is not clear how to model objectives b–e by means of linear functions. However, if we do not want to optimize the functions in those objectives, but just impose lower and upper bounds on them, we can do this with linear constraints. For example, if we want the average slab weight to be at least ten tons, we can insist that the total weight of mother plates designed is at least ten times the number of mother plates used. For each of the objectives, we imposed a bound on the objective function value using soft or hard constraints based on an analysis of the expected objective values. In our application, the surplus ratio was expected to be 3% or lower, and we imposed an upper bound of 3% on the surplus ratio as a hard constraint. We imposed lower bounds on the values of the other objective functions with soft constraints and penalized the violation of these constraints. For example, since we expected more than 90% of the rush orders to be completed, we imposed the condition that the total number of completed rush orders plus an integral slack variable is at least 0.9 times the total number of rush orders. We then penalized noncompletion of at least 90% of rush orders by using a nonzero coefficient for the above slack variable in the model objective.

In addition to penalizing violation of the above objective bounds, we also have an objective function coefficient, or *score*, for each pattern based on how good the pattern is with respect to each of the five objectives. This score is a function of the weight of the mother plate, the surplus weight on it, the associated waste, the fraction of orders used in the mother plate that are completed by the order plates therein, and the fraction of rush orders completed by rush-order plates on the mother plate.

We create an integer program with integer variables x_j corresponding to one- or two-dimensional patterns of order plates (the variables count the number of times a pattern is used). *A pattern* is one possible way of arranging order plates on a mother plate. We impose the usual cutting stock constraints: The variables x_j are non-negative, the total number of designed plates of an order cannot exceed the maximum demand for the order, and the total capacity of any facility used for the designed order plates cannot exceed the facility capacity. For rolling mills, the total weight of mother plates processed in them must lie in a range. In addition, we introduce non-negative integer variables z_i that count the number of multiples of a charge weight that can be designed for a grade set. Suppose the orders in a pattern have a grade g, and g belongs to five different grade sets. We create five copies of this pattern and assign each copy a distinct grade set. We then add the constraint that the total weight of patterns for a given grade set i roughly equals the mean charge weight times z_i.

Let A stand for the matrix of patterns, i.e., A_{ij} gives the number of plates of order i in pattern j. Let B be a matrix in which B_{ij} equals the weight of pattern j if it has grade group i, and 0 otherwise. Let D be a matrix such that D_{ij} gives the weight of orders in pattern j that need processing on facility i. If facility i stands for a rolling mill, D_{ij} gives the weight of the mother plate if the route of the mother plate includes that rolling mill, and 0 otherwise. Let d_L and d_U stand for the lower and upper bounds on the demand for an order. Let the vectors L and U stand for the lower and upper bounds on the weight that can be processed at the different facilities. The components of L are zero except for the rolling mills. Let diag(d_L) and diag(d_U) stand for the diagonal matrices with d_L and d_U arranged on the diagonal. Let y_c and y_u be as described before. Let w, s, and v stand, respectively, for the vectors of pattern weights, surplus weights, and waste weights. Assume that there are m orders. Assume that we want

the bounds on requirements a–e to be $k_a, \cdots, k_e$, and let $s_a, \cdots, s_d$ stand for the penalty terms associated with requirements a–d. Let $\vec{1}$ stand for a vector with all components equal to one of appropriate dimension. Then $\vec{1}^T x$ stands for the sum of the variables x_i. The integer program we solve is this:

$$\max c^T x + \vec{1}^T z - s_a - s_b - s_c - s_d ,$$

subject to the constraints

$$Ax \le d_U ,$$

$$\mathrm{diag}(d_L) y_c \le Ax \le \mathrm{diag}(d_U) y_u ,$$

$$-25 \le Bx - 275z \le 25,$$

$$L \le Dx \le U,$$

$$\sum_{i \in O_R} (y_c)_i + s_a \ge k_a ,$$

$$\vec{1}^T y_c + s_b \ge k_b \vec{1}^T y_u ,$$

$$(w - v)^T x + s_c \ge k_c w^T x,$$

$$w^T x + s_d \ge k_d \vec{1}^T x,$$

$$s^T x \le k_e w^T x,$$

$$x, z \ge 0,$$

x, z integral,

and

$$y_c, y_u \in \{0, 1\}^m .$$

A natural question is why we use two different ways of handling the same objectives, i.e., why we have mother-plate scores that take into account surplus weight in addition to the surplus ratio constraint. The bounds in the soft and hard constraints associated with requirements a–e are only estimates based on data for multiple days, but may not be suitable for a particular day's data. Suppose that an average surplus ratio of 3% is expected, but on a particular day a solution with a surplus ratio of 1% is easily obtained. In such a case, the surplus ratio constraint plays no role, and a solution of the MPD problem could be trivially non-optimal with respect to surplus ratio. For example, a solution with a surplus ratio of 1% could be made worse by taking a mother plate and adding a surplus plate on it. Without mother-plate scores, the second solution would be treated as equal to the first.

As in the usual CSP, we (approximately) solve the linear relaxation of the above integer program with delayed column generation. We call the integer program above with the entire set of patterns the *master IP*, and the associated linear relaxation the *master LP*. We call the IP (LP) defined by a partial set of patterns the *partial master IP* (LP). We start off with an initial set of patterns (or, more precisely, the copies of patterns with assigned grade sets), the above constraints, and the variables relaxed to be real numbers. The column-generation subproblem consists of taking the optimum dual solution of the partial master LP and finding one or more negative reduced-cost patterns. The partial master LP is then augmented with columns corresponding to the negative reduced-cost patterns.

In our application, we have up to 5,000 orders. It is clear that solving the master IP exactly is not feasible in a reasonable amount of time, and we solve it only approximately.

Column generation

To solve the column-generation subproblem (CGP), we iterate through every combination of order component, target route, and order width. Recall that mother-plate lengths depend on the width of the mother plate (which equals maximum order width for simple patterns), and also on the target route. For each such combination, we compute l_{min} and l_{max} (based on the target slab geometry) and then find negative reduced-cost patterns and add them to the master problem. Traditionally in cutting stock, the CGP for simple patterns is modeled as a knapsack problem. This is easy to do if the objective function coefficient for each pattern in the master problem is a linear function of the order plates, surplus plates, and waste in the pattern, and the width and length of the mother plates are fixed. Unfortunately, the objective function coefficient for each pattern in our application is a nonlinear function, and the usual knapsack approach does not work.

Fortunately, most of the order components have very few orders (two or three), with a few components having up to 50 orders. Thus, for most of the components, simple heuristics suffice. We use the following packing heuristics for simple patterns:

- KPOG (knapsack with fixed orientation and guillotine cuts): We assign weights to the orders and, for each order as the maximum-width order in a mother plate, we solve a knapsack problem. The objective function value of a solution to this knapsack is an approximation of the reduced cost of the corresponding pattern.
- BFD (best-fit decreasing): We simply sort all orders on the basis of the dual prices of orders and whether or not orders are rush orders, and apply the BFD bin-packing heuristic to create bins.

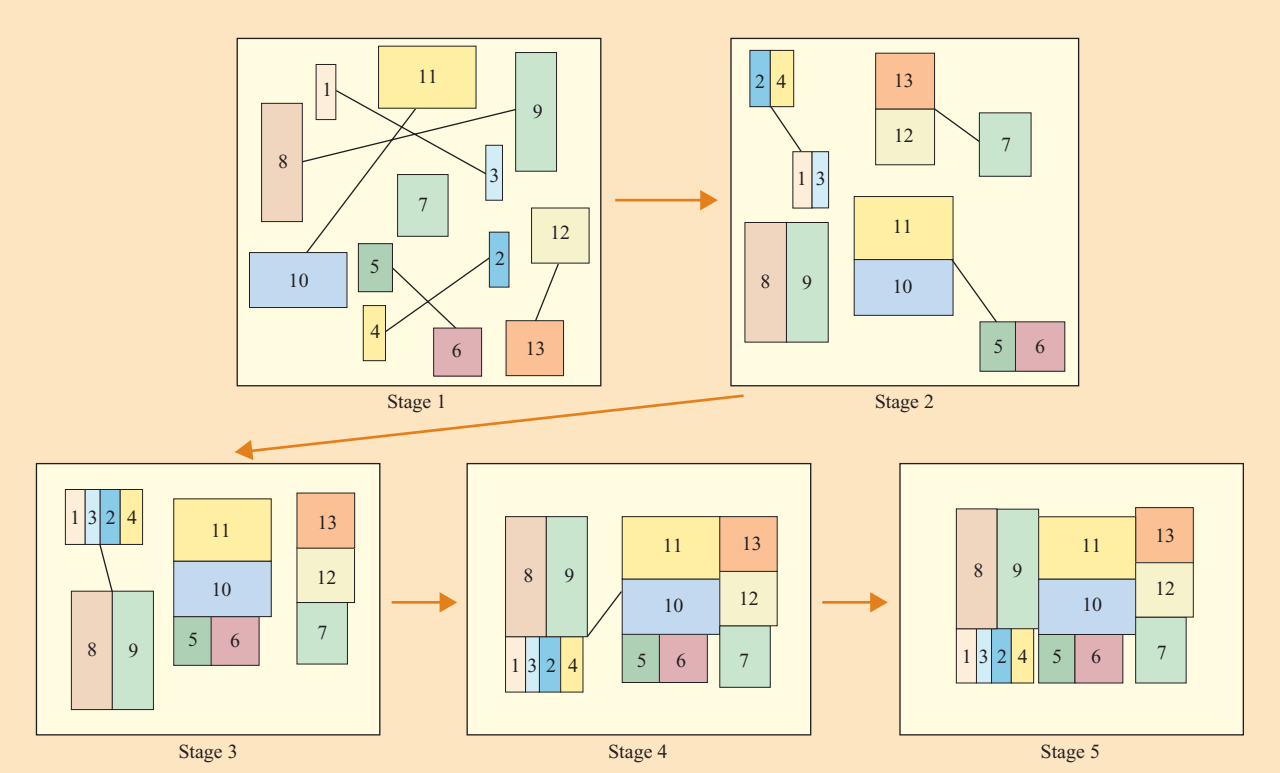

Figure 4

Five stages in the operation of the NBWM algorithm (colored rectangular boxes are order plates).

For mosaic patterns, we use the following heuristics:

- FBS (first-best strip): See [12]. Many other algorithms from the literature can be adapted to generate feasible mosaic patterns.
- NBWM (non-bipartite weighted matching algorithm): See [13].

Figure 4 illustrates the operation of the NBWM algorithm in the context of mosaic pattern generation. Groups of order plates placed adjacent to each other represent partially built mother plates. A line (or edge) joining a pair of order plates or partial mother plates indicates that they can be packed together on a mother plate. We assign such an edge a numeric weight that reflects the desirability of packing the orders or partial mother plates together. In each stage, we calculate a maximum weight matching of partial mother plates, join partial mother plates to get larger partial mother plates, and continue until one or more complete mother plates are available (stage 5).

Orders have dual values assigned to them by the master problem. Typically, in initial invocations to the CGP, we make these values identical for all orders, but later we obtain them from the partial master LP, and thus they vary by orders. When the dual values are identical, the objective of the CGP is to create feasible mother-plate patterns (which are also desirable with respect to various criteria, such as the amount of wasted metal on the mother plates). When the dual values are not identical, the CGP involves creating mother plates such that the reduced cost of the mother plates is as small as possible. We designed dual-value-sensitive variations of our heuristics by sorting on dual values of orders in FBS or BFD or incorporating dual values into weights for KPOG or NBWM. We do not explicitly use the dual values associated with the capacity and grouping constraints in our CGPs, but we do use them in testing whether or not the columns returned by the subproblems have negative reduced cost.

Solving the master IP

Various authors have solved CSPs by combining the delayed column-generation scheme to get LP bounds along with enumeration using a branch-and-bound tree [14, 15], though usually with only a few hundred orders (see also [16]). Our approach to solving our variant of the

CSP can be viewed as diving down the branch-and-bound tree to some leaf. We repeatedly do the following until we have an acceptable integer solution to the MPD solution. We perform delayed column generation until our partial master LP solution value changes minimally from one iteration to another, then solve the partial master IP. Clearly this IP solution may not be a good approximation to the optimal master IP solution. To improve the quality of this solution, we fix a part of the IP solution; i.e., we set the values of a number of variables to their IP solution values. We then use the fixed variables to update the right-hand sides and get a modified integer program.

Invocations of the MPD problem

The MPD subproblem is invoked many times by the CD module. The different invocations can be grouped into two distinct types: grade grouping and template filling.

In the grade-grouping invocation, the CD module invokes the above subproblem first with no restriction on the slab widths corresponding to the designed mother plates (the CD width exploration phase), and then later with the slabs restricted to specific widths. In either case, the MPD module returns mother plates that can be partitioned in different grade groups, with the weight per grade group being an approximate multiple of the mean charge weight. In these invocations, usually only a partial list of orders is given as input. Further, these invocations are for the design of slabs on a specific caster. Because these invocations are for partial order books, where the expected values of the different objective criteria are not known, the mother-plate scores are the primary determinant of the objective function.

With regard to the template-filling invocation, recall that templates are descriptions of the charge layout in casts, i.e., the charge-grade groups in a cast and their sequence. When the CD module has made a final decision regarding the cast templates, it removes all slabs from these cast templates and invokes the MPD module with a precise count of the number of charges to be designed for each grade group. This type of invocation is performed only once, as opposed to grade-grouping invocations. This can be handled conceptually by fixing the variables z_i for the different grade groups.

Once a collection of mother plates have been designed, a set of possible slab geometries is created for each mother plate, as discussed above in the section on slab design.

Inventory allocation

The inventory consists of surplus plates, slabs, and cast slabs. The geometry of a surplus plate is inflexible, since all dimensions (thickness, width, and length) are fixed. Further, surplus plates are relatively small, which severely restricts the orders that can be cut from them. Usually only a single order plate can be cut from a surplus plate. Slabs have moderate design flexibility, since the associated MPD is not fixed. Cast slabs obviously have higher design flexibility compared with slabs, because the sizes of slabs on a cast slab can be chosen. One can vary the number of slabs cut from cast slabs as well. The inventory allocation module of the PD tool returns a list of slabs to be cut from cast slabs, a list of mother plates to be made from the slabs, and the location of order plates on the mother plates.

The inventory allocation problem can be viewed as a variant of the multiple-knapsack problem or as a generalization of the 2D CSP. In the traditional multiple-knapsack models in the steel industry, the decision problem is to allocate orders directly to the existing stock materials while maximizing the allocated profits (order weights). Vasko et al. [17], Kalagnanam et al. [18], Dawande et al. [19], and Forrest et al. [20] studied the allocation problem of coil products in the steel industry and proposed heuristics including matching and bin packing. The inventory allocation problem we tackle is different because of the dimension transformations from slabs to mother plates. For example, in the cast-slab allocation problem, the cast-slab materials are cut into multiple bins (slabs) that are rolled into mother plates and then cut into orders.

The main goal in the inventory allocation problem is to maximize the weight of order plates designed from inventory items while minimizing waste. The way we model and solve this problem is very similar to our approach for the MPD problem. We decompose the problem into a master problem, which selects from candidate allocation patterns for each inventory item (surplus plate, slab, or cast slab) and a subproblem that generates these patterns. An *allocation pattern* is an MPD compatible with the geometry of the inventory item. For example, an allocation pattern for a plate or slab is a candidate mother plate, while that for a cast slab is a collection of candidate mother plates, each of which corresponds to a slab, with the total weight of mother plates in the pattern equal to the cast-slab weight. In a sense, we have one subproblem per inventory item type.

There are two main differences with respect to the MPD problem. The first is that only one allocation pattern can be selected per inventory item, whereas a pattern can be repeated any number of times in mother-plate design. The second difference is that the grouping constraints are not present in inventory allocation. As in mother-plate design, we assign a score generated by a complex calculation to each allocation pattern. The score information is calculated as the weighted sum of multiple attributes, such as due dates, allocated order weights, and yield rates of the allocation patterns. Note that during

pattern generation, orders can belong to multiple allocation patterns. Order feasibility is resolved by the master LP and IP.

Assume that we have generated multiple candidate allocation patterns for each inventory item. The master problem selects patterns by solving the integer programming formulation CGMaster defined as

$$\text{maximize} \sum_{k \in K} \sum_{m \in CAP(k)} c_{km} \cdot x_{km} \tag{1}$$

$$\text{subject to} \sum_{k \in K} \sum_{m \in CAP(k)} A_{kmi} x_{km} \leq (d_U), \ \forall i \in O, \tag{2}$$

$$\sum_{m \in CAP(k)} x_{km} \leq 1, \ \forall k \in K, \tag{3}$$

$$x_{km} \in \{0, \ 1\}, \tag{4}$$

and some additional constraints.

Here K stands for the set of inventory items, $CAP(k)$ is the set of candidate allocation patterns for the inventory item k, and $x_{km} = 1$ if the mth allocation pattern for item k is selected, and 0 otherwise. Here A_{kmi} stands for the number of plates of order i in the mth allocation pattern for item k, and c_{km} stands for the cost of the mth allocation pattern for item k. Constraint (2) restricts the number of plates of an order that can be allocated from inventory to, at most, the upper bound on the demand for order i. Constraint (3) states that only one candidate allocation pattern can be selected per inventory item. If we know all of the feasible candidate allocation patterns beforehand, an optimal integer solution of CGMaster defines the best collection of feasible allocation patterns for the inventory materials.

The overall algorithmic flow implies that the success of the algorithm strongly depends on generating good allocation patterns within a reasonable time. The pattern-generation subproblem for inventory plates is quite simple: Check whether an order plate can be cut from an inventory plate given the associated steel grades and geometries. For an inventory slab, we design mother plates compatible with the geometry of the slab using the same ideas and pattern-generation code used in the pattern-generation subproblem of the MPD problem. The cast-slab allocation subproblem is a bit more complicated. We need to decide how to cut an inventory cast slab into slabs and then how to generate mother plates compatible with these slabs. Our approach is to generate mother-plate patterns compatible with the cast-slab width and thickness (slabs cut from a cast slab have the same width and thickness), and then to combine mother plates to form a cast-slab allocation pattern using a simple bin-packing algorithm.

Cast design

The CD problem is to design casts for the input orders so that daily capacities are met and production constraints are satisfied. In addition to the SD objectives, some of the objectives emphasized during cast design are maximizing the rush-completion ratio, minimizing the surplus-slab ratio, and maximizing the average number of charges per cast.

The four main steps in cast design—candidate cast design, cast selection, cast template determination and filling, and cast-slab design—are shown in **Figure 5** and described in the following sections.

Candidate cast design

In this step, several candidate casts are designed for the given input order plates. This is done iteratively, once for each combination of caster and mold thickness. Each iteration starts with the determination of good widths at which to design casts to fulfill as many rush orders as possible while keeping the weight of surplus slabs or plates low. This is followed by the design of candidate casts at each of the selected widths.

Cast-width determination

At certain cast widths, only mother plates with a lot of surplus plates or waste can be generated. Discarding such poor-quality mother plates entails designing many surplus slabs to satisfy the charge weight constraint and requiring a minimum number of charges per cast. Even if mother plates with minimal waste or surplus can be designed, there may not be enough rush weight, i.e., the weight of slabs whose constituent orders are rush orders. We also want other slab design metrics such as average slab weight to have desirable values.

First, a set of slabs is designed while optimizing the metrics of the SD objectives by solving an SD problem with grade-grouping constraints but no dimension constraints. Note that each of the slabs has a range of feasible widths and a slab grade. Using the grade-transition property, as given by the graph GT, the set of grades for the set of slabs is divided into grade components. On the basis of grade components, a slab component is constructed for each grade component by collecting slabs belonging to the grades in the grade component. Note that no cast can contain slabs from different slab components. Grades in each grade component correspond to the union of mixable grade sets in H.

For each grade set S_i present in a grade component, slabs whose slab grade is in S_i are collected in a slab set. Slabs in a slab set can be put together in a charge. The width ranges of the slabs in a slab set are represented by an interval graph. By using a polynomial algorithm [21], maximal cliques of the interval graph are determined. The

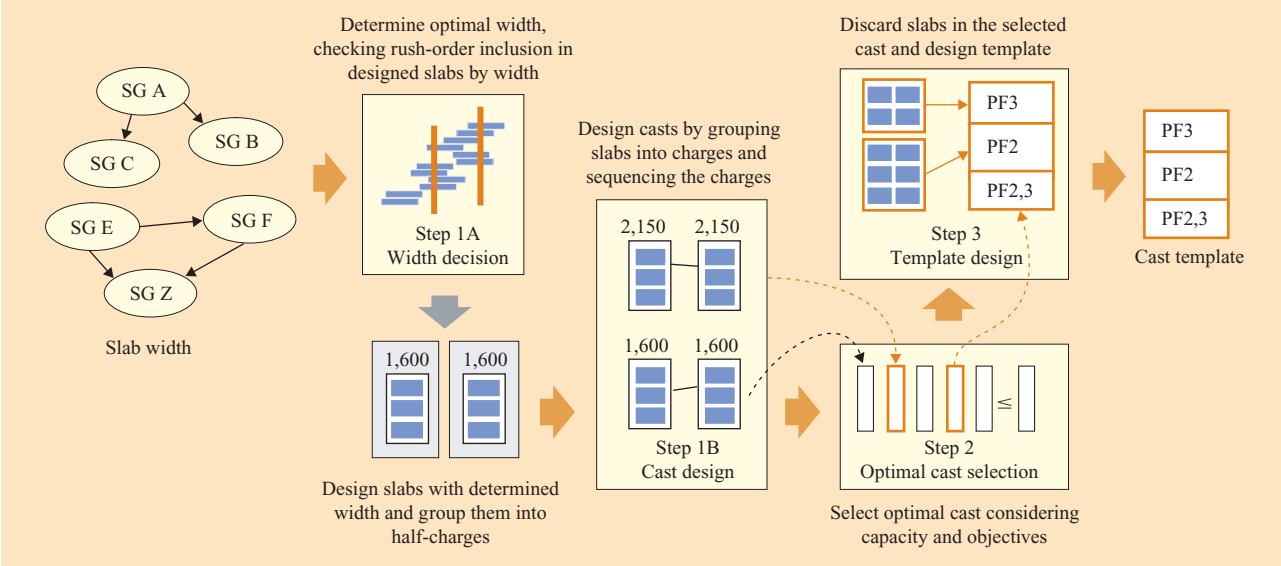

Figure 5

Overview of cast design.

slabs are collected for each of the maximal cliques. This provides us with sets of slabs, each of which is related to a grade-width clique. Note that a slab can belong to more than one clique. By using a set-packing formulation with the objective of minimizing surplus within a charge, grade-width cliques are selected so that a slab belongs to, at most, one grade-width clique. This formulation was solved using ILOG CPLEX** 9.0.

So far, we have grade-transition components, each of which has a set of grade-width cliques. Each grade-width clique c_i contains slabs that can go into a charge and have a common width range, r_i. Let r^* be the width range spanning all r_i. Then, for each width $w \in r^*$ such that w is divisible by 10, the sum of rush-potentials of all of the slabs present in the grade-width cliques whose common width range contains w is computed. The rush potential of a slab is defined as max(0, *rush-order weight in a slab − surplus in the slab*). The width that provides the most total rush-potential (for all grade-transition components) is selected as the width at which slabs are designed and cast. This step is repeated `MaxWidthIterations` times—each time excluding the widths that are selected in the previous iterations. The value of `MaxWidthIterations` is determined experimentally.

For each of the choices of width, the SD module is given the chosen slab width as input. Further, slab design is also given the capacity of each rolling mill. The rolling mill capacity is determined by using the capacity limits for each rolling mill multiplied by an external parameter, `capacityMultiplier`. The value of this parameter indicates the amount of flexibility that should be given (consequently, the amount of runtime allowed) to the CD phase. The set of slabs supplied by this SD step are next used for designing candidate casts.

Design of candidate casts from slabs

The purpose of this step is to design several candidate casts based on the given set of slabs for the subsequent step of cast selection. This step contains three substeps: making half-charges, pairing half-charges into charges, and sequencing charges into casts.

The first step in cast design is building charge-strands (or half-charges, because the casts in this problem are restricted to have exactly two strands). For the given set of slabs, grade-width cliques are determined in the same way as described in the previous step of selecting widths, except for where the interval-graph-based maximal cliques are determined (because all slabs are trivially in a maximum clique with the width range specified by the chosen width). The list of slabs in each grade-width clique is sorted by an externally specified sorting function. This list is chopped into sublists, where each sublist contains slabs whose cumulative weight is no more than half the maximum charge weight.

The next step is to pair the set of half-charges into charges such that the variation of attributes, e.g., the number of grades in a charge and the number of possible refinement routes, is minimized. This is accomplished by computing a max-weight non-bipartite matching on a

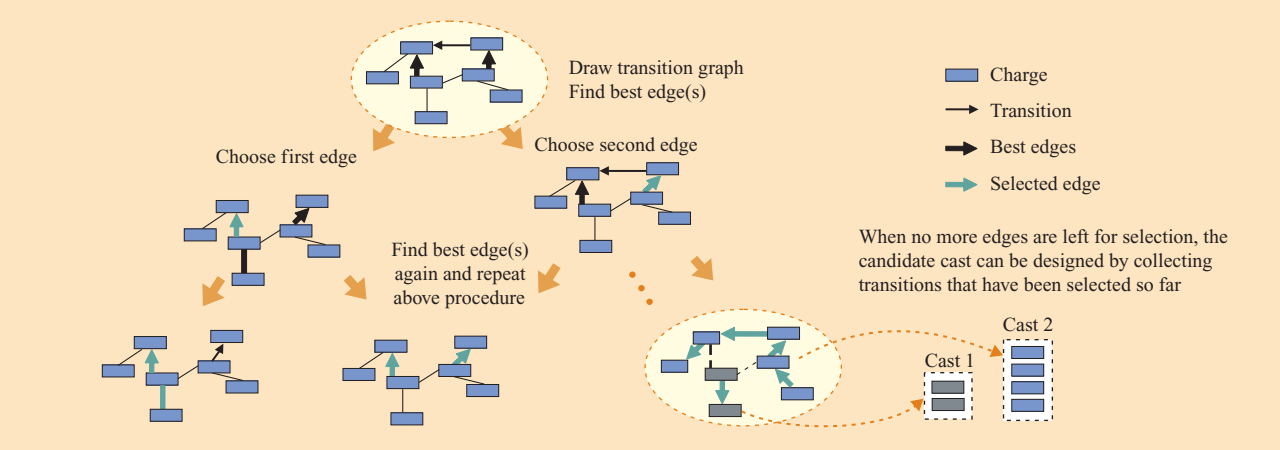

Figure 6

EBB algorithm for cast design.

graph with half-charges as nodes, edges between all pairs of nodes, and an edge weight on each edge that is inversely proportional to the variation of attributes in the charge that results from pairing the half-charges corresponding to the vertices of the edge.

CD problem

At this point, a cast is basically a sequence of charges with two additional restrictions. The first restriction is that the number of charges in a cast should be within a range $[l_g, u_g]$ that is dependent on the grade set g of the constituent charges and the caster. The second restriction is that the grades of any two consecutive charges must be compatible according to the grade-transition graph. The min limit l_g on the number of charges is a soft constraint in the sense that when the number of charges in a cast n is below l_g, the cast can be appended with $(l_g - n)$ surplus charges. The CD problem is to build casts for k charges from a given set of charges so that (in the following order of priority) 1) the quantity (*rush weight* − *surplus weight*) is maximized; 2) the number of charges per cast is maximized; and 3) the number of grade transitions between charges per cast is minimized. In addition, the objectives are calculated over c best casts so that the sum of numbers of charges in these c casts is at least k, and the sum of numbers of charges in the top $c - 1$ casts is less than k.

Charge sequencing

The algorithm used to solve the CD problem is called the extended branch-and-bound (EBB) heuristic. (For a detailed description, see [22].) EBB starts with a solution in which each charge is a cast fragment. Different possible ways of sequencing cast fragments result in different solutions. The decision on whether two fragments can be merged depends on CD constraints. EBB searches among various possible solutions (**Figure 6**). Among the current possible solutions, the best m solutions are chosen on the basis of the objectives, where m is a parameter external to the algorithm. Among the best solutions, each possible merge is evaluated, and the top b best fragment merges for each solution are chosen, where b is the beam width, which is specified as a parameter to the algorithm. The best merge is the one that results in the maximum value of (*rush weight* − *surplus weight*) per charge among all possible merges; ties are broken on the minimum number of grade transitions per charge. This is similar to the multifragment method for constructing initial solutions in solving traveling salesman problems [23]. The crucial difference here is that up to b choices of merges at each level are simultaneously explored. Note that each fragment is a linear sequence of charges with no forks. Again, the best m solutions in the resulting $b.m$ solutions are chosen. This continues until no more merges are possible or the time limit, which is specified as an external parameter, is reached. EBB finally returns the m best solutions.

A cast is created that corresponds to each fragment in each solution. If the number of charges in a cast is less than l_g, surplus charges are created up to l_g and appended to the cast.

It turns out that sometimes creating long casts may not be a good idea for reasons related to capacity. If the remaining capacity of a resource is c charges, but the demand of a cast on that resource is more than c charges, that cast cannot be chosen, however good it may otherwise be. However, a part of the cast whose demand on the resource is no more than c can be chosen. To take

care of such situations, cast solutions with shorter casts are created by running EBB r times, where r is the difference between the maximum $NCC_{\max}$ over all grades and widths and the minimum l_g over all grades and widths. In the ith run $0 \leq i \leq r$, each fragment is limited to being no bigger than $\max(l_g, u_g - i)$ charges.

Cast selection

The plant has daily capacity limits for each resource, such as casters, rolling mills, and refining stations. Some of these capacities are in the number of charges and some are in tons of weight.

Among all of the casts designed in various solutions through the various runs of EBB, the cast-selection step chooses casts such that no resource capacity is exceeded and no order plate is present in more than one cast, while maximizing on the objective defined in the following way. The objective function coefficient of cast i is composed of the rush potential of the cast (*rush weight − surplus weight*), denoted as C_i, and the cumulative objective value of slabs in cast i, as calculated during slab design, denoted as D_i.

This is accomplished using a set-packing formulation with side constraints to enforce capacity limitations. Let I stand for the set of designed casts, and let the decision variable x_j denote whether cast $j \in I$ is selected. Let A be a matrix such that A_{ij} equals the number of plates of order i present in cast j, and let D be a matrix such that D_{fj} stands for the weight of slabs in cast j that require processing on facility $f \in F$.

$$\text{Maximize} \sum_{j \in I} r_j d_j x_j$$

subject to

$$\sum_{j \in I} A_{ij} x_j \leq 1, \quad \text{for all orders } i \in O,$$

$$\sum_{j \in I} D_{fj} x_j \leq U_f, \quad \text{for all facilities } f \in F,$$

$$x_j \in \{0, 1\}, \ \forall j \in I.$$

Cast-template determination and filling

For each of the selected casts, a cast template is constructed. A cast template consists of charge templates corresponding to the charges in the cast. Each charge template contains the salient information about the corresponding charge, such as the set of grades in the charge, the width and the thickness of the slabs in the charge, the set of rolling mills to which the slabs in the charge are assigned, the refinement processes a charge uses, and the weight of the charge. The charge-template specifications are grouped and indexed by tuples of mixable-grade sets, width, thickness, rolling mill, and refining process. For each template-specification tuple, the corresponding list of charges and their weights are collected.

This information is passed to slab design for designing slabs. The output of slab design (template-filling mode) is a set of slabs for each specification. One by one, each of the charges corresponding to the specification is filled up to its specified weight using the slabs from the set of slabs. If the weight of a charge falls below the minimum, the charge is filled with surplus slabs that are constructed using one of the slabs in the charge as a replica.

Cast-slab design

All slabs in a cast slab must go to the same rolling mill. Moreover, the total weight of cast slabs in a charge should be within the minimum and maximum charge weight limits. The number of cast slabs in a charge strand of a charge can differ from that of the other charge strand in the charge by at most 1. There is also a limit on the maximum allowed difference in length of charge strands in a charge. At the boundaries of charges of different grades and at the beginning and the end of casts, because of production issues such as grade mixing and degradation, there are constraints on the grades of cast slabs that can be placed in these areas. We solve this problem for each charge separately using heuristics and MIP formulations, where the problem of combining the smaller slabs into cast slabs and the problem of selecting cast slabs to satisfy charge-level constraints are solved simultaneously.

Computational issues and results

A crucial limitation imposed by the real-life setting of our application is that there are numerous constraints that can only be approximately modeled either because they are too complex or because their precise description is not known. An example of the latter is the entire set of feasibility constraints on mother-plate patterns. Every mother-plate pattern is checked for manufacturing feasibility (taking into account issues such as the rolling and cutting precision) by a complex and time-consuming software application at the client's site, which we treat as a black box. The approximate model of this black box given to us allowed us to generate mother-plate patterns that were feasible at least 95% of the time, but not always. A time limit of 30 minutes was imposed on our code, which runs on a desktop PC with an Intel P6 CPU and 4 GB RAM. Because a substantial portion of that time is taken up by the feasibility-checking black box, the extent of column generation we can perform to solve the MPD subproblems is restricted. Further, the very large number of constraints implies that in some cases there are no

Table 1 Percentage of improvement in objectives as a result of template filling.

Objectives	*Improvement (%)*
Rush-completion ratio	22.3
Orders completed	13.7
Average slab weight	3.7
Surplus ratio	0.0
Yield ratio	−0.2

Table 2 Comparison of PD tool results with results using prior method.

Objectives	*PD tool Setting 1* (% improvement)	*PD tool Setting 2* (% improvement)
Rush-completion ratio	19.0	−1.7
Average slab weight	8.7	6.5
Surplus ratio	−17.9	−3.6
Yield ratio	1.4	1.5
Surplus-slab ratio	−39.0	54.0

reasonable alternatives to simple heuristics or exhaustive enumeration.

We use a combination of MIP models and heuristics to solve the PD problem. We use CPLEX 9.0 to solve the linear programming problems arising in different contexts, and also a number of MIPs. For the MIPs, we use different parameter settings depending on the specific subproblem. We solve about four to six MPD problems with grouping constraints (one with no slab width specified during width exploration, and the rest with specified slab widths) for each of the 15 or so combinations of casters and thicknesses, for a total of about 50 to 100 MPD problems. This implies a time limit of about 30 seconds per subproblem of this type, which is why we do not combine a branch-and-bound enumeration with column generation. For these problems, we are satisfied with approximately optimal solutions, and we also use specialized rounding heuristics that produce solutions whenever CPLEX cannot produce a solution within the prescribed time limit. The EBB heuristic runs quickly, and the set-packing formulation for cast selection is relatively easy to solve to exact optimality, since we usually generate at most a few hundred casts before selecting from five to ten cast columns from them.

The template-filling problem (mother-plate design with grade groups fixed) is fairly time-consuming. We spend up to a third of our budgeted time on this problem, but this is justified because of the overall improvement in the quality of the casts. In **Table 1**, we illustrate the percentage of improvement in various objectives after template filling is performed using the cast templates from the final selection of casts. It is based on one specific data set containing 3,815 orders with a total weight of about 38,000 tons, of which 626 of the orders are rush orders. The solution after template filling in Table 1 consists of eight casts and 720 mother plates (the number of casts does not change in template filling). The rush-completion ratio is the ratio of the number of completed rush orders to the total number of rush orders. For reasons of confidentiality, we cannot provide the exact values of the different objectives before and after template filling, but we give a range for these values. Before template filling, around 1,000 orders and between 40% and 60% of the rush orders are completed. The average slab weight lies between eight and ten tons, the yield ratio is more than 85%, and the surplus ratio is at most 5%.

We are unable to give an exact comparison of the values of the important objectives before and after the deployment of our PD tool at our client's plant, because the production process was changed in anticipation of efficiency gains resulting from the use of the tool. For example, since our tool could generate solutions with a higher average number of charges per cast, the minimum number of charges per cast was increased when our tool was used.

We are, however, able to present a comparison of solutions obtained by the PD tool with a solution obtained using the prior (semiautomated) method used by our client (**Table 2**). The data set consisted of 2,315 orders with a total weight of about 25,000 tons and 723 rush orders. The objectives are the basically the same as those in Table 1, except that we do not give the order-completion ratio; instead, we give the surplus-slab ratio in the last row. For Setting 1, we give the percentage of improvement in objective values, as compared with the client solution, by setting a high emphasis on completing rush orders. The solution for Setting 2 is based on a different set of parameters. In the first case, the PD tool yields a solution with many more completed rush orders and higher slab weight, but with more surplus plates and slabs. In the second case, the PD tool returns a solution that trades off reduced surplus slabs for reduced rush-order completion.

Summary

This paper describes the use of optimization and analytic tools to improve production design for plate products in the steel industry. The use of these tools provides two types of benefits: improvement in the productivity of a plant and an approach to incorporate the key business

performance indicators (such as available-to-promise) into operations at a production level.

In our experience, the productivity gains achieved at the plant level are of two types: improved yield and reduced surplus in the design of slabs and casts. Depending on the product mix and the grade mix, our PD tool is effective in reducing the surplus to 3% to 5% of the total production (measured in terms of the weight of the designed slabs) while increasing the average slab weight by about 3% to 5%. In tandem, yield improvements are gained by increasing the average number of charges per cast. This could improve the yield by up to 0.5%. For a one-million-ton plant, these improvements could result in direct cost savings of more than $2 million annually.

Another benefit of PD tools developed here is the ability to integrate supply-chain-level planning (based on a six-month horizon) to day-to-day scheduling (to the level of ten minutes) that is feasible on the plant floor. Since the production-design approach presented in this paper is able to manage due dates for orders in conjunction with such operational measures as yield and surplus rates, it provides an explicit way to incorporate key business-performance indicators (such as available-to-promise and productivity) into operations. For example, when customer satisfaction is the highest priority (Setting 1 in Table 2), it provides a way to improve on-time delivery (a strategic measure that is reflected in the rush-completion ratio) by trading off a little on the operational measure of surplus. Setting 2 provides a way to maximally improve operations (such as the surplus-slab ratio) while achieving a desired rush-completion ratio.

As manufacturing companies turn their attention from planning to incorporating their plans into operations to provide measurable benefits, optimization tools for production design will play a key role in facilitating this transition. This will have a tremendous impact on steel companies and other manufacturing-centered industries.

Acknowledgments

We thank Mark Trumbo for his help in implementing a part of our solution approach, and Yong Kwan Kim, SangJo Kim, and ShinKyu Kwak for their help in communicating with our clients and clarifying the problem to be solved. We also thank the reviewers for their helpful suggestions.

**Trademark of ILOG, Inc. in the United States, other countries, or both.

References

1. G. Dutta and R. Fourer, "A Survey of Mathematical Programming Applications in Integrated Steel Plants," *Manuf. & Service Oper. Manage.* **3**, No. 4, 387–400 (2001).
2. O. Porto, F. Menezes, L. Moreno, and E. Uchoa, "Optimization of the Continuous Casting Phase in Steel Tubes Production," *Proceedings of the 19th International Symposium on Mathematical Programming*, Rio de Janeiro, Brazil, 2006; see *http://www.ismp2006.org/Informacoes/detailedprogram.php*.
3. I. Harjunkoski and I. E. Grossman, "A Decomposition Approach for the Scheduling of a Steel Plant Production," *Computers & Chem. Eng.* **25**, No. 11, 1647–1660 (2001).
4. S. Y. Chang, M.-R. Chang, and Y. Hong, "A Lot Grouping Algorithm for a Continuous Slab Caster in an Integrated Steel Mill," *Production Planning & Control* **11**, No. 4, 363–368 (2000).
5. D. Pacciarelli and M. Pranzo, "Production Scheduling in a Steelmaking–Continuous Casting Plant," *Computers & Chem. Eng.* **28**, No. 12, 2823–2835 (2004).
6. H. S. Lee, S. S. Murthy, S. W. Haider, and D. V. Morse, "Primary Production Scheduling at Steelmaking Industries," *IBM J. Res. & Dev.* **40**, No. 2, 231–252 (1996).
7. P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem," *Oper. Res.* **9**, No. 6, 849–859 (1961).
8. P. Gilmore and R. Gomory, "A Linear Programming Approach to the Cutting Stock Problem—Part II," *Oper. Res.* **11**, No. 6, 863–888 (1963).
9. F. Vanderbeck, "A Nested Decomposition Approach to a Three-Stage Two-Dimensional Cutting-Stock Problem," *Manage. Sci.* **47**, No. 6, 864–879 (2001).
10. S. Michel, N. Perrot, and F. Vanderbeck, "Knapsack Problems with Setups," Working Paper No. U-04.03, Department of Applied Mathematics, Université Bordeaux, Talence Cedex, France, 2007; see *http://www.math.u-bordeaux.fr/~fv/papers/fcknpPap.pdf*.
11. M. A. Vonderembse and R. W. Haessler, "A Mathematical Programming Approach to Schedule Master Slab Casters in the Steel Industry," *Manage. Sci.* **28**, No. 12, 1450–1461 (1982).
12. A. Lodi, S. Martello, and D. Vigo, "Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems," *INFORMS J. Computing* **11**, No. 4, 345–357 (1999).
13. A. Fritsch and O. Vornberger, "Cutting Stock by Iterated Matching," *Proceedings of the International Conference on Operations Research*, Berlin, Germany, 1995, pp. 92–97.
14. Z. Degraeve and L. Schrage, "Optimal Integer Solutions to Industrial Cutting Stock Problems," *INFORMS J. Computing* **11**, No. 4, 406–419 (1999).
15. Z. Degraeve and M. Peeters, "Optimal Integer Solutions to Industrial Cutting-Stock Problems: Part 2, Benchmark Results," *INFORMS J. Computing* **15**, No. 1, 58–81 (2003).
16. D. L. Applegate, L. S. Buriol, B. L. Dillard, D. S. Johnson, and P. W. Shor, "The Cutting-Stock Approach to Bin Packing: Theory and Experiments," *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments (ALENEX)*, Baltimore, MD, 2003, pp. 12–15.
17. F. J. Vasko, D. D. Newhart, and K. L. Stott, "A Hierarchical Approach for One-Dimensional Cutting Stock Problems in the Steel Industry that Maximizes Yield and Minimizes Overgrading," *Euro. J. Oper. Res.* **114**, No. 1, 72–82 (1999).
18. J. R. Kalagnanam, M. W. Dawande, M. Trumbo, and H. S. Lee, "The Surplus Inventory Matching Problem in the Process Industry," *Oper. Res.* **48**, No. 4, 505–516 (2000).
19. M. Dawande, J. Kalagnanam, H. S. Lee, C. Reddy, S. Siegel, and M. Trumbo, "The Slab-Design Problem in the Steel Industry," *Interfaces* **34**, No. 3, 215–225 (2004).
20. J. J. H. Forrest, J. Kalagnanam, and L. Ladanyi, "A Column-Generation Approach to the Multiple Knapsack Problem with Color Constraints," *INFORMS J. Computing* **18**, No. 1, 129–134 (2006).
21. U. I. Gupta, D. T. Lee, and J. Y.-T. Leung, "Efficient Algorithms for Interval Graphs and Circular-Arc Graphs," *Networks* **12**, No. 4, 459–467 (1982).

22. H. S. Lee and M. Trumbo, "An Approximate 0–1 Edge-Labeling Algorithm for Constrained Bin-Packing Problem," *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997, pp. 1402–1411.
23. J. L. Bentley, "Experiments on Traveling Salesman Heuristics," *Proceedings of the 1st Annual ACM–SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 1990, pp. 91–99.

Received October 2, 2006; accepted for publication December 14, 2006; Internet publication May 11, 2007

Sanjeeb Dash *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (sanjeebd@us.ibm.com)*. Dr. Dash is a Research Staff Member. He received a Ph.D. degree in computational and applied mathematics from Rice University in 2001 and worked as a Postdoctoral Fellow at Princeton University before joining the IBM Research Division in 2002. His research interests lie in the area of discrete optimization, especially mixed-integer programming. He has worked on many aspects of cutting-plane techniques for solving mixed-integer programming problems. Dr. Dash is a coauthor of the linear programming solver QSopt, and of QSopt_ex, an exact linear programming solver.

Jayant Kalagnanam *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (jayant@us.ibm.com)*. Dr. Kalagnanam is a research manager in the Mathematical Sciences Department. He holds a Ph.D. degree from Carnegie Mellon University and joined the IBM Research Division in 1996. The focus of his research is manufacturing sciences. He has worked with various steel companies to develop and deploy optimization and scheduling solutions for daily plant operations. Dr. Kalagnanam has more than 20 journal publications and more than ten patents and filings.

Chandra Reddy *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (creddy@us.ibm.com)*. Dr. Reddy joined IBM in 1998 after receiving a Ph.D. degree in computer science from Oregon State University. Since then, he has been developing solutions to several optimization and scheduling problems arising in the steel industry. Several of these solutions have been successfully deployed and are in production at two of the world's leading steel makers. His research prior to joining IBM focused on both theoretical and practical aspects of machine learning of compact hierarchical control knowledge for planning and problem solving. Dr. Reddy's research interests include scheduling and optimization, machine learning for planning and classification, data mining, and artificial intelligence.

Sang Hwa Song *Graduate School of Logistics, University of Incheon, Gyeonggi Province, Korea (songsh@incheon.ac.kr)*. Professor Song received a Ph.D. degree from the Department of Industrial Engineering at the Korea Advanced Institute of Science and Technology in 2003. He worked for the IBM Korea Technology Laboratory and IBM Business Consulting Services as a senior software engineer and senior consultant. Professor Song's research interests include supply chain management and an application of optimization theories in real-world business problems.