RC25065 (W1010-096) October 21, 2010
Computer Science

IBM Research Report

Scheduling Time-Critical Flowshops: An Example from the Steel Industry

Fatima Kilinç-Karzan
H. Milton Stewart School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Jayant R. Kalagnanam, Andrew J. Davenport, Stuart Siegel, Chandra Reddy
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Scheduling time-critical flowshops: An example from the steel industry

Fatma Kılınç-Karzan
H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, fkilinc@gatech.edu

Jayant R. Kalagnanam, Andrew J. Davenport, Stuart Siegel and Chandra Reddy
IBM T.J. Watson Research Center, Yorktown Heights , NY 10541, jayant@us.ibm.com, davenport@us.ibm.com, stus@us.ibm.com, creddy@us.ibm.com

Production in metals and chemicals industry imposes tight maximal time lag constraints between some process stages, leading to time-critical flowshop problems that are characterized by constraints restricting the work-in-progress inventory at some stations to be effectively zero. We study a complex industrial scheduling problem arising from steel manufacturing with time-critical flowshop structure. Our core problem has a flowshop structure with multiple modes and recipes, sequence dependent setup times, strict precedence constraints between certain jobs, and inventory considerations.

We present a novel approach to solve this class of problems by decomposing it into stages and integrating integer programming (IP) with constraint programming (CP) through logic-based Benders decomposition. We enhance our technique with a special class of aggregate resource use inequalities, obtained through a detailed constraint programming based schedule for each *job*. We further strengthen our formulation through the polyhedral study of an important common substructure, single machine scheduling with sequence-dependent setup times problem. Our solution method has been successfully deployed in the daily operations of a large integrated steelmaker with improvements in ontime delivery from 75% to over 90%.

Key words: Production Scheduling; Steel Industry; Time-indexed Formulations; Logic-based Benders Decomposition

1. Introduction

Time-critical flowshops are characterized by a tight constraint on the wait time for any job at given station. When the wait time is significantly less than the processing time on that station, this implies that the work-in-progress (WIP) inventory at the station (buffer) to be zero. If each process in the flow shop had a single machine then this would lead to a *permuation schedule*. However in the general case of each process having more than one machine, the sequence across different processes does not need to preserved. Traditionally, the ability to incur WIP inventory at each stage of the flowshop provides a convenient way to decouple the scheduling between different processes. Each stage of the flowshop can be solved independently as long as the processing of the activities of a job are temporally ordered, at the cost of incuring large WIP inventories. In time-critical flowshops,

the no WIP inventory constraint tightly couple all the processes of the flowshop and thereby makes it very difficult to find feasible schedules satisfying productivity constraints.

Time critical flowshops are commonly encountered in the metal and chemical processing industry. In the hot metal shop, molten metal is being processed and hence it cannot be allowed to wait between processing steps since this leads to crust formation. In the chemical industry, the processing steps consist of purification or chemical reactions. Wait time between processes is not allowed since this could impact the reaction or the constituents of the batch.

We have chosen to present our method for the more general time critical scheduling problem through the study of hot metal shop in large steelmaker in the Asia Pacific region. Presenting formulations in a real life setting clarifies the motivations for the chosen models as well as makes it easier to follow, thus provides a more succinct characterization of the problem while being easy to transfer to other settings.

1.1. Overview of Steel Making Operations

The production of steel is composed of 3 primary steps - the hot metal shop where the molten iron is transformed into slabs, the rolling mill where slabs are rolled into plates, coils or billets, and the finishing mill where the coils, and plates are galvanized and tempered to meet appropriate characteristics (e.g., corrosion resistance and shape) required by the order. In the literature, steel making and continuous casting processes are usually identified as the bottleneck processes in steel production (see Cowling and Rezig (2000), Tang et al. (2002)). Our experience in this project supports this assertion. Therefore, our focus in this paper is on modeling and optimization of the operations for steel making and continuous casting in the hot metal shop of a steel plant. The scope of the scheduling problem we address covers the production of solid steel slabs from molten iron. The goal is to produce a multi-day operational level schedule for the manufacture of steel products from raw material inputs that trade off productivity against customer satisfaction objectives (i.e., on time completion).

In the hot metal shop, molten iron from the Blast Furnace (BF) is refined in the Basic Oxygen Furnace (BOF) and Refining Processes (RF) in batches of about 250 to 300 metric tons (MT), each batch resulting in steel with different metallurgical properties. At the casting stage, these batches of refined molten steel are poured through a mold with rectangular cross section on a continuous caster (CC). The strand of metal emerging from a mold is cut across its length to form a cast *slab*, which is again cut lengthwise to form additional (sub) slabs of steel that typically weight 15-25 MT each. We refer to a sequence of slabs created from the same mold as a *cast*, and a batch of steel and the slabs produced from it as a *charge*. A cast can thus be viewed as a

sequence of charges. Since 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. These requirements, as well as constraints on how the width and thickness can differ between successive slabs, impose detailed sequencing restrictions for each cast and its charges. Furthermore, once the processing of a cast on the continuous caster is started, all of its charges are required to be processed in a pre-specified order without any time gap in between them. The batching of orders into charges and the sequencing of charges into casts are provided as input to our scheduling system. These batching and sequencing steps are tackled as complex, multi-criteria optimization problems in and of themselves, the descriptions of which are provided in Dash et al. (2007) and are outside of the scope of this paper.

1.2. Solution Approach

The scheduling model for a hot metal shop can be viewed as a flow shop scheduling problem, however there are some significant variations that arise from (i) the batch treatment of hot metal, (ii) strict precedence constraints between some jobs on certain machines, (iii) the extremely stringent limits on the waiting times (maximal time lags) between processes, and (iv) hot metal inventory constraints. In addition, there are sequence dependent setup times, shift level capacity constraints and target requirements on the amounts of certain types of products to be produced.

In this paper, we model the steel production in the hot metal shop as two strongly connected manufacturing processes: upstream (basic oxygen furnace and refining) and downstream (continuous casting). The upstream processes are scheduled on unary capacity resources but have multiple available routes and alternate machines for each process operation. The downstream problem involves the selection and sequencing of casts on casters subject to shift-level capacity constraints and target requirements on the number of orders of each product type, tight inventory constraints and sequence-dependent setup times arising from costly reconfiguration of the casters. Upstream and downstream processes are connected through tight minimum and maximum waiting time constraints to ensure the temperature restrictions of the steel between processes.

We present a time-indexed scheduling formulation which models the decisions regarding the downstream process (the selection of casts and their planned start times on the casters) in full detail. Our efforts to capture full details of the individual charges in a Mixed Integer Programming (MIP) formulation lead to unmanageable sizes in terms of both number of variables and constraints, due to the enormous number of the charges (around 400) to be modeled, the flexibility of processing route in the upstream processes for each charge, and the required time granularity of

30 seconds in the operations in upstream processes. On the other hand, Constraint Programming (CP) techniques are very effective for solving feasibility problems especially with tight constraints (see Kanet et al. (2004)). In our case, the downstream problem imposes strict temporal constraints in terms of time windows for the start time of charges on the upstream processes, hence constraint filtering techniques are very effective in performing domain reduction and thus solving the upstream feasibility problem. Moreover, our CP model, the details of which is presented in Davenport et al. (2007), is capable of taking into account detailed scheduling constraints in a time granularity of even 30 seconds and preferences on different routes for the processing of an individual charge.

This simple approach of decomposing the problem into two processes (upstream and downstream) leads to an important drawback : often downstream production plans can be generated which do not lead to feasible upstream schedules satisfying the waiting time requirements and inventory constraints. Therefore, we develop a logic-based Benders decomposition approach which uses mixed integer programming to generate a production plan for the downstream processes (master problem), and constraint programming to generate a detailed feasible schedule for the upstream processes (sub-problem). In order to enhance the strength of the Benders decomposition, we include additional inequalities to represent the estimations of upstream capacity utilization in the downstream scheduling problem. This approach turns out to be very effective in providing high quality schedules that routinely outperform the hand-generated expert schedules on both productivity and delivery time based objectives.

We apply this method to a scheduling problem arising in the steel manufacturing industry with the objective of maximizing on-time delivery while keeping productivity at desired high levels. The solution method has been successfully deployed in the daily operations of a large integrated steelmaker with improvements in on time delivery from 75% to over 90%.

1.3. Layout of the Paper

The paper is organized as follows. We first provide a detailed description of the operations and constraints of steel manufacturing in the hot metal shop in Section 2. In Section 3, we review the literature on time indexed formulations for scheduling, steel related scheduling, and logic based Benders decomposition. We present our formulation for the downstream scheduling problem together with the polyhedral study of single machine scheduling with the sequence dependent setup time problem in Section 4. We develop a logic-based Benders decomposition which is used to connect our downstream and upstream models in Section 5. We discuss the impact of our approach on the current daily operations of the company and report computational results that outline the performance of our approach for a set of problem instances taken from real life in Section 6. Finally, we discuss the effectiveness of this approach and give future research directions in Section 7.

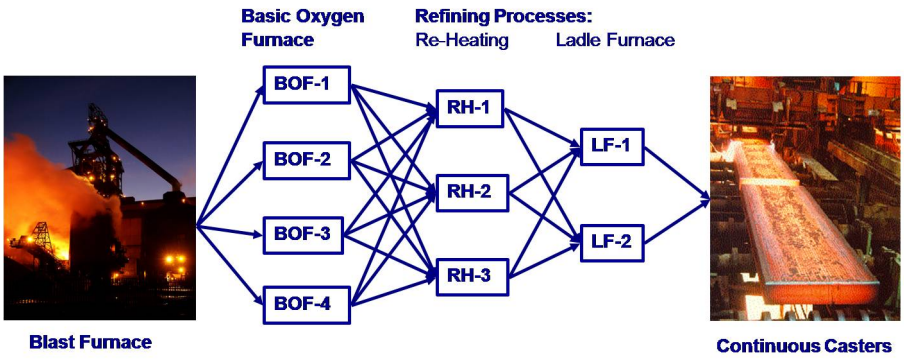

Figure 1 **Hot metal shop in a steel making plant**

2. Description of the Steel Making Operations

In this section, we discuss in detail the main aspects of operations performed in hot metal shop illustrated in Figure 1. The main processes involved with the manufacturing of steel slabs can be categorized as follows:

1. *Blast Furnace* (**BF**) The production of steel starts at the blast furnace where iron is heated to a very high temperature to become molten. There is a continuous flow of molten iron from the blast furnace to the downstream processes. The hourly rate of the flow is given as input to the scheduling problem.

2. *Basic Oxygen Furnace* (**BOF**) is the first process that all production must pass through after leaving the blast furnace. This is where the molten iron starts to become differentiated with respect to grade and chemical composition. The processing here is done in batches of a few hundred MT and a batch is referred as a *charge*.

3. *Refining Processes* (**RH, LF**) consist of a number of steps such as reheating, ladle furnace and stirring. The steps that are required and their ordering depend on the chemical composition of the final products, therefore it's not necessary for all batches to pass through these steps.

4. *Continuous Casters* (**CC**) is the final stage of production at the hot-metal shop and it is required for all products. In this process, molten steel is poured into a long, adjustable copper mold. As the steel passes through the mold, it is cooled by water jets and it solidifies and then is cut into slabs of specific dimension. The production of slabs on a caster is done in batches referred as *casts*.

All distinct operations at the basic oxygen furnace and the refining stages take place on a single charge of steel. However at the continuous casting stage, operations take place on a cast (a sequence of charges, typically 4-12 in number, to be processed contiguously on the caster).

2.1. Objectives of the Schedule

Steel is usually produced on a make to order basis. As new orders arrive, production plans are revised and based on the anticipated production plans, due date promises are negotiated with the customers. When orders for the steel slabs are received, based on a production plan, they are assigned a planned start time (PST), typically specified at the granularity of a shift or a day, for each of the process steps in hot metal shop, rolling mill and finishing mill. In order for actual production to meet due dates and keep customer service levels at an acceptable level, each process shop (and its process operations) is required to honor the PST assigned to the slabs.

In our models, for a given cast, we take the minimum of all the PSTs of its constituent slabs and use this as the PST for the cast. We assign a score to each cast inversely proportional to the delay between the scheduled date and the PST. Notice that in this way the casts that are overdue (i.e. the schedule date is already past the PST of the slab) are treated as rush orders and given a (slightly) higher priority. The objective for the scheduling engine in this setting is to maximize the total score of the casts produced for any given time horizon.

2.2. Production of a Single Cast

We present a Gantt chart showing a schedule for the operations involved in the production of a single cast of steel in Figure 2. The significant aspects of this model are:

- Each activity is non-preemptible and performed on a single charge of steel, i.e., $A1$, $A2$, $A3$ and $A4$ in Figure 2 are performed consecutively on the same charge of steel.

- In the final process, casting, a cast is produced by contiguously processing a pre-specified sequence of charges without any interruptions and time gaps. In Figure 2, the cast is composed of charges A, B and C in this order, and the casting operations of these charges are $A4$, $B4$ and $C4$. Hence on the caster, the start time of operation $B4$ must occur at the end time of operation $A4$, and the start time of operation $C4$ must occur at the end time of operation $B4$.

- There are tight minimal and maximal time lag constraints between consecutive activities performed on the same charge. The constraints on the maximum waiting time arise as a result of the temperature requirements of molten steel, i.e., if the temperature drops below a certain point due to long waiting, it's necessary to reheat it, which is expensive in terms of energy consumption. This is so critical that if the delay exceeds a certain time limit, the molten iron will freeze, destroying the torpedo car. These maximum waiting time constraints reflect the requirements on the temperature of steel and they are different for each charge. Minimum waiting time constraints arise from the transfer time of molten steel between processes. For instance the maximum (minimum) waiting time between the end of $A1$ and start of $A2$ is 20 (5) minutes, in Figure 2.

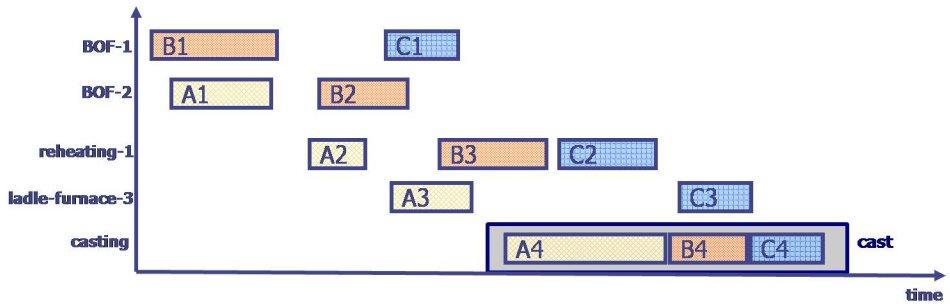

Figure 2 **A Gantt chart illustration of the operations (activities) involved in manufacturing a single cast in a steel plant, from the basic oxygen furnace (BOF) to the refining processes (reheating (RH), and ladle furnace (LF)) to the casting (CC) process.**

- At each process, there are a number of non-uniform parallel resources that can be used to perform the same operations. The different operating characteristics and physical locations of these parallel resources lead to non-uniformity. As a result, the processing time of an activity of a process depends on which resource it is assigned to, as well as the transfer times between processes.
- Each charge comes with a preferred recipe specifying a sequence of process steps that it must pass through. We are also given between 0 and 3 alternate recipes that can be used in the case of insufficient capacity on the process stages of the preferred recipe. In practice, most (85-95%) of the charges are expected to be assigned to their preferred recipes.
- All the scheduling resources in the problem have unary capacity (at any time, at most one activity can be executed on a specific resource). However, some resources have an associated state, where this state is represented by the number of activities processed since the last "setup" activity on the resource. Once the resource has processed a maximum number of activities, it is required to perform a setup activity, which resets the state to zero. Some activities can only be processed on the resource if the state of the resource is less than a certain number. This is a strict requirement for refining furnaces (RF) and hence the availability of a RF needs to be tracked in terms of its state.

2.3. Model of Cast Scheduling

In section 2.2 we presented the basic scheduling model for the production of a single cast. In the full problem we are required to select and schedule between 50 to 100 casts, each composed of 1-8 charges, out of a larger (120) set of casts, on a number (3-9) of distinct casting machines taking into account all the constraints imposed by the individual cast model over a 1-2 day horizon. The planning horizon is decomposed into shifts (a period of 8 hours) and there are 3 shifts per day.

In this model, our goal is, to maximize the number of charges that are processed on their preferred dates by selecting and determining which casts to schedule and at what times in a shift. Recall

that the specified order of charges in a cast and the assignment of casts to the casters are part of the input. Other specifications related with full model can be listed as follows:

- *Setup times:* There are sequence dependent setup times between consecutive casts processed on the same casting machine. The setup time between two casts depends on the similarity between their casting mold requirements. A cast following another one would need a mold change if (i) the grade of the two casts is incompatible, and/or (ii) the width difference between the last slab of the previous cast and the first slab of the next cast is substantial. In the case of a setup, the time needed is based on the time to adjust the mold from the previous shape to the new shape which depends on cast sequence.
- *Capacity constraints:* Shift level capacity constraints are specified to ensure high utilization of available resources while maintaining desired load balance for downstream processes (e.g. rolling mill and finishing mill). Each charge has attributes such as product type and grade. These constraints state the minimum and maximum number of charges that can be produced per shift by each attribute and are specified in two flavors:

  — A total target for the number of charges scheduled for each shift (or day) to ensure minimum productivity at the hot melt shop, and,

  — Target number of charges for each product type (coil, plate and bloom) for load balancing of the rolling mills. As different products feed slabs into separate downstream rolling mills (e.g. rolling plates is different from rolling coils), it is important to provide sufficient slabs to each rolling mill to ensure continuous operation.
- *Hot metal inventory constraints:* The continuous flow of molten iron from the blast furnace is stored in torpedo cars, and then transported to the basic oxygen furnace. These torpedo cars provide a finite capacity inventory buffer for the storage of molten iron until it is consumed at BOF to form charges. Having full torpedo cars queuing up in front of BOF increases their turnaround times, results in fewer torpedo cars available for the draining of the blast furnaces. This in turn leads to a reduction in the production rate of the blast furnaces, degrading the molten iron and in the worst case causing damage and lengthy down time. Hence the number of available torpedo cars leads to maximum inventory level constraints on the quantity of molten iron that can be allowed to accumulate in this buffer. On the other hand, insufficient allocation of molten iron will cause the steelmaking operation to shut down and the downstream production will be afflicted with significant cost penalties. Hence there are also minimum inventory requirements to ensure enough iron feeding the steelmaking process. Therefore balancing the flow of iron from the blast furnace with the consumption of iron feeding into the steelmaking and casting processes is essential

to improve the overall efficiency and reduce energy consumption. These constraints are considered on a 1 hour time bucket granularity. Since this is a rolling horizon model, an initial inventory of iron in the torpedo cars is also specified.

Note that the presence of inventory constraints eliminates the opportunity of using regular scheduling formulations with continuous start time variables. However, this important aspect can easily be represented with time-indexed variables of appropriately chosen time granularity. Therefore our focus in the rest of the paper is on scheduling formulations based on time-indexed variables.

3. Literature Review

Scheduling problems arising from steel industry are recognized among the most difficult scheduling problems as they involve multi-stage operations tightly connected with inventory and waiting time constraints. Therefore there have been several efforts to model and solve various stages of scheduling problems occurring in steel industry. In Lee et al. (1996), an overview of the process and computerized scheduling solutions including OR-techniques, AI methods and combinations of these two are provided. Tang et al. (2001) provide an excellent survey on the general steel production planning problems. Although most of the available methods for overall steel manufacturing systems use expert systems (Dorn (1996), Dorn and Shams (1996)); heuristics (Missbauer et al. (2008)); fuzzy logic or auction-based (Kumar et al. (2006)) approaches to generate feasible solutions, there are also several optimization studies focusing on only parts of the overall system. Among the optimization studies, Lally et al. (1987) and Tang et al. (2000) develop mixed integer models for continuous casting process alone. Moreover Tang et al. (2002) propose a Lagrangian relaxation based heuristic algorithm to solve the steel making problem. Harjunkoski and Grossmann (2001) develop a decomposition approach for scheduling of a steel production plant, which is later on extended to multistage in Harjunkoski and Grossmann (2002). Cowling and Rezig (2000) provide a model and solution approach for the integration of caster scheduling and hot strip mill planning. Tang et al. (2007) model the allocation of molten iron from blast furnaces to the steel making furnaces as a parallel machine scheduling problem with time windows. After reformulating the problem as a set packing problem, they propose a Branch and Price algorithm. Although the setting of Tang et al. (2007) is closest to ours, they don't model the continuous casting operations explicitly.

In the literature, there have been many studies on scheduling. Lawler et al. (1993) presents a good overview of scheduling literature. The reader is further referred to Queyranne and Schulz (1994) for an excellent survey on the polyhedral study of scheduling problems. Since the results in

this paper are focused around time-indexed formulations, next, we provide a general overview of time-indexed formulations in the scheduling literature.

Time-indexed formulations are well-known for their tight LP-relaxations although they suffer a lot from their extensive sizes in terms of number of variables and constraints. van den Akker et al. (1999), Crama and Spieksma (1996), Sousa and Wolsey (1992) and Waterer et al. (2002) discuss the polyhedral study of time-indexed formulations and present valid inequalities for various scheduling problems. Sousa and Wolsey (1992) consider a single machine scheduling problem, show the dimension of the polytope and provide 3 classes of valid inequalities based on Generalized Upper Bound technique. Crama and Spieksma (1996) work on a special case of single machine scheduling problem with release dates and equal processing times, study its polyhedral structure and prove that this problem is NP-hard even when all the processing times are 2 units. van den Akker et al. (1999) also study the polyhedral structure of single machine scheduling problem and provide implicit characterizations of all the facets with right hand sides 0, 1, and 2. Waterer et al. (2002) consider the relation of time-indexed formulation for single machine scheduling problem to the node packing polytope. They show that the results provided in van den Akker et al. (1999) can be proved through this relation. The study of Hardin et al. (2008) focuses on time-indexed formulations for resource constrained scheduling problems where they develop valid inequalities and lifting procedures. Finally, it is important to note that even obtaining the solution of the LP relaxation of a time-indexed formulation can be quite expensive hence there have been research on improving the LP-relaxation solution times for time-indexed formulations by column generation approach (see van den Akker et al. (2000)). Recently Pan and Shi (2007) establish the equivalence of the best lower bound obtained from the transportation relaxation and the LP relaxation of time-indexed formulation for the single machine scheduling problem with min-sum type of objective.

The scheduling problems in real life are fairly complicated due to constraints relating different decision levels resulting in very large quite hard to solve models. To overcome this difficulty, logic-based Benders decomposition is introduced by Hooker and Yan (1995) in the context of logic circuit verification. This approach extends the usual Benders approach by allowing the sub-problems to be not only a linear program but also any other from which can be represented by logical constraints. The approach partitions the variables into two classes. After deciding the values of the variables in the first class through the solution of a MIP master problem, a sub-problem is solved for the second class variables by fixing the values of the first class variables and applying constraint programming techniques.

In related work, Hooker (2006) applies logic-based Benders method to the problem of minimizing tardiness on parallel machines. Chu and Xia (2005) solve the minimum makespan problem by generating Benders cuts derived from an IP formulation of the sub-problem. Logic-based Benders methods have also been adapted to solving various IP problems (Hooker and Ottosson (2003), Chu and Xia (2004)) and the propositional satisfiability problem (Hooker (2000), Hooker and Ottosson (2003)). Moreover Jain and Grossmann (2001) successfully apply logic-based Benders to minimum cost planning and scheduling problems where the sub-problems are single machine scheduling problems. Harjunkoski and Grossmann (2002) extend this work to multistage problems by decomposing the problem into assignment of jobs to the machines and sequencing phases. Similar ideas have been applied to minimal dispatching of automated guided vehicles (Corréa et al. (2004)), steel production scheduling (Harjunkoski and Grossmann (2001)), real-time task scheduling of computer processors (Cambazard et al. (2004)), batch scheduling in a chemical plant (Maravelias and Grossmann (2004)), and polypropylene batch scheduling (Timpe (2002)). In all these applications except Hooker and Ottosson (2003) and Chu and Xia (2004), the sub-problem is a feasibility problem. Similar Benders type approaches have been proposed in pure constraint programming context where both master and subproblems are solved through constraint programming techniques (see Cambazard et al. (2004), Eremin and Wallace (2001)).

Note that logic-based Benders decomposition requires repeated solves of the master problem with Benders cuts derived at each iteration. Considering that the master problem itself is a MIP, even a single solve of the master problem can be very expensive. To overcome this difficulty, Hooker (2000) proposes *Branch and Check* algorithm. In *Branch and Check*, Benders cuts are generated for each integer feasible solution found during the solution process of the master problem hence allowing the master to be solved only once. With the implementation of Branch and Check algorithm, in Thorsteinsson (2001), an additional speedup is reported over the results of Jain and Grossmann (2001). Branch and Check is also successfully applied to the problem of minimizing the sum of the weights of late jobs of a single machine scheduling problem with release dates in Sadykov (2008).

4. Mathematical Formulation

We first give our notation for problem parameters included in the integer programming formulation and then introduce our basic model for cast scheduling. We extend this formulation by estimating and including the capacity requirements of upstream resources in aggregate form and modeling hot metal inventory constraints. In addition to this, we provide a polyhedral study of a common substructure in our model and propose some valid inequalities to enhance the solution of our IP formulation.

Table 1 Notation

T:	time horizon
I:	set of all casts
C:	set of available casters
s_i:	score of cast i, i.e., objective function coefficient of x_{it}
c_i:	caster on which cast i can be processed
p_i:	processing time of cast i on the assigned caster
R_{ij}:	setup time required when cast j is processed immediately after cast i on the caster
$J(c)$:	set of casts assigned to caster c
$\{1,\ldots,h_i\}$:	set of charges of cast i ordered with respect to their processing order on the caster
B:	set of available BOF machines
P:	set of all product types
d_{ip}:	number of charges of cast i with product type p
$N_p^{min}(N_p^{max})$:	minimum (maximum) production amount for product type p
$M^{min}(M^{max})$:	total minimum (maximum) production amount in terms of number of charges
ρ_t:	hot metal inventory amount transferred from BF to BOF at the beginning of period t
I_{-1}:	initial hot metal inventory amount
δ_i^j:	amount of hot metal inventory consumed by j^{th} charge of cast i at BOF processes
$K_t^{min}(K_t^{max})$:	minimum (maximum) amount of inventory that can be kept in the torpedo cars during time period t

For simplification in the representation of constraints in our formulation, we assume that $R_{ij}=0$ when there is no setup requirement between casts i and j.

4.1. Basic Downstream Formulation

We start with providing our basic time-indexed formulation representing only the scheduling of casts on the casters and then extend this model to include other previously stated complicating constraints (capacity, hot metal inventory, etc.) as well as the aggregated constraints coming from the modeling of the upstream processes. We assume that the time horizon and processing and setup times are transformed into the equivalent discrete time intervals used in the model.

We let

$$x_{it} = \begin{cases} 1, & \text{if the processing of cast } i \text{ is started at the beginning of time period } t \\ 0, & \text{otherwise} \end{cases}.$$

where $t \in \{0,1,\ldots,T-p_i\}$ for all $i \in I$. Then using these time-indexed decision variables, our formulation becomes:

$$\max \sum_{i\in I}\sum_{t=0}^{T-p_i} s_i x_{it} \tag{1}$$

$$\sum_{t=0}^{T-p_i} x_{it} \le 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i\in J(c)} \sum_{s=\max\{t-p_i+1,0\}}^{\min\{t,T-p_i\}} x_{is} \le 1 \qquad \forall t \in \{0,1,\ldots,T-1\},\ \ \forall c \in C \tag{3}$$

$$x_{it} + \sum_{s=\max\{t-p_j-R_{ji}+1,0\}}^{\min\{t+p_i+R_{ij}-1,T-p_j\}} x_{js} \le 1 \qquad \forall t \in \{0,1,\ldots,T-p_i\},\ \ \forall i,j \in I \tag{4}$$

$$N_p^{min} \le \sum_{i\in I}\sum_{t=0}^{T-p_i} d_{ip} x_{it} \le N_p^{max} \qquad \forall p \in P \tag{5}$$

$$M^{min} \leq \sum_{i \in I} \sum_{t=0}^{T-p_i} h_i x_{it} \leq M^{max} \tag{6}$$

$$x_{it} \in \{0,1\} \qquad \forall t \in \{0,1,\ldots,T-p_i\}, \quad \forall i \in I \tag{7}$$

Constraints (2) state that each cast can be started at most once on the assigned caster. Constraints (3) indicate that at any given time, on each caster, only one cast can be processed. The setup requirements between the casts processed on the same machine are represented through (4). The production requirements on individual product types and the total production requirements are given as constraints (5) and (6) respectively.

Although this model fully captures the scheduling requirements of casts on the casters, it does not represent individual charges and their operations performed at the upstream processes. Next, we discuss derivation of constraints regarding the upstream processes for our IP model through CP techniques.

4.2. Schedule Estimation for Upstream Processes

As stated in Section 2, all the charges have to be processed on BOF machines and casters, however the processing requirements on the refining processes varies. Moreover due to the availability of alternative recipes for refining processes, it turns out that the capacity requirements on BOF machines are a lot tighter than the ones in refining processes and it is usually quite beneficial to model the BOF capacity requirements for the upstream processes in the downstream formulation.

In order to develop an aggregate representation of the upstream processes in terms of the cast start time variables x_{it}, we first define the parameter $\hat{S}^j_{it}$ ($\hat{F}^j_{it}$) as the estimated start (finish) time of j^{th} charge of cast i on a BOF machine if the processing of cast i is started at the beginning of time interval t, i.e., $x_{it} = 1$. Based on the BOF capacity requirements of each charge, a BOF capacity profile is built for each cast start time variable, x_{it}, see Figure 3.

Unfortunately the parameters $\hat{S}^j_{it}$ and $\hat{F}^j_{it}$ are not readily available to our models. Different characteristics and processing requirements of individual charges and the availability of multiple processing routes make the estimation process rather dynamic. On the other hand the start time of a cast i on the caster defines a time interval for the start time of each charge $k = 1,\ldots,h_i$ on the upstream machines. Using CP techniques, we compute the upstream schedule for each cast individually to estimate $\hat{S}^j_{it}$ and $\hat{F}^j_{it}$. Basically for any given cast start time value, the constraint programming solver tries to find a feasible upstream schedule satisfying the setup and processing requirements on the machines and waiting time constraints for all the charges associated with the cast. The details of these constraint programming techniques are discussed in Davenport et al. (2007).

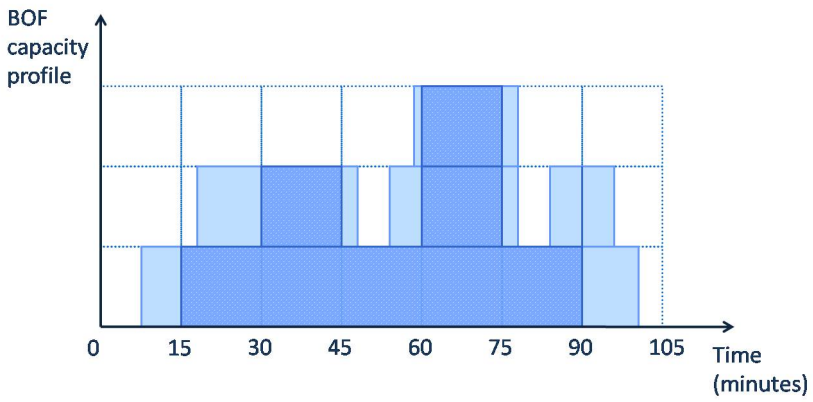

Figure 3 **An illustration of the estimation of BOF profile for the charges of a single cast.**

Another difficulty in the upstream schedule estimation is due to different time granularity required for detailed modeling of upstream and downstream processes: the time granularity for cast sequencing formulation is around 15-30 minutes whereas the time granularity required for the upstream processes is 30 seconds. Because of this large variation in time granulations and the dynamic estimation process, the resulting intervals $[\hat{S}^j_{it}, \hat{F}^j_{it}]$ available in the capacity profile usually do not coincide with our time period epochs. In order to convert these into the time granularity used for downstream formulation, depending on the purpose of the estimation, one can choose to underestimate or to overestimate the capacity requirements for each cast. To underestimate (overestimate), the time of each increase event in the capacity profile, i.e. $\hat{S}^j_{it}$, have to be shifted to the start of the first (last) time period after (before) the event's actual time and the time of each decrease event, i.e. $\hat{F}^j_{it}$, in the capacity profile have to be shifted to the start of the last (first) time period before (after) the actual event time. In order to add constraints with more flexibility regarding the upstream processes, we choose to underestimate the BOF capacity requirements for each cast. Let $[S^j_{it}, F^j_{it}]$ denote the interval obtained after underestimation, then $[S^j_{it}, F^j_{it}] \subseteq [\hat{S}^j_{it}, \hat{F}^j_{it}]$. In Figure 3, we represent the estimated BOF profile for a cast with light region and the corresponding underestimated BOF profile is given by the dark region.

Additionally, observe that for some cast and start time combinations (x_{it}), it is quite possible that the estimated capacity profile is beyond the limits of time horizon, i.e., the first charge of the cast has to start processing on BOF before the beginning of time period 0, $\hat{S}^0_{it} < 0$. In those cases, we trust our capacity profile estimations and assume that we cannot start the processing of cast i at time t as the upstream schedule of its charges is infeasible, therefore we set $x_{it} = 0$.

Using the estimated capacity profiles, we can write the BOF capacity constraints in the aggregated form as follows:

$$\sum_{i\in I}\sum_{s=0}^{T-p_i}\sum_{\substack{j\in\{1,\dots,h_i\}:\\ t\in[S_{is}^j,F_{is}^j]}} x_{is} \le |B| \qquad \forall t\in\{0,1,\dots,T-1\} \tag{8}$$

Finally, in order to have a feasible production plan, we need to model the requirements on the hot metal inventory as discussed in Section 2.3. Recall that the incoming flow of molten iron from BF is stored in torpedo cars and later consumed when the processing of a charge starts at BOF. Clearly, this establishes the other essential use of our upstream resource estimation. At the beginning of each time period, a flow balance equation is written to capture the inventory restrictions. For notational purposes, we introduce the following new variables:

$$I_t = \text{ the amount of hot metal inventory at the beginning of period } t$$

where I_{-1} denotes our starting inventory. Together with our assumption that the inventory is consumed at the beginning of a period if a charge starts being processed on a BOF machine, we write the inventory balance constraints as follows:

$$I_t = I_{t-1} + \rho_t - \sum_{i\in I}\sum_{s=0}^{T-p_i}\sum_{\substack{j\in\{1,\dots,h_i\}:\\ S_{is}^j=t}} \delta_i^j x_{is} \quad \forall t\in\{0,1,\dots,T-1\} \tag{9}$$

$$K_t^{min} \le I_t \le K_t^{max} \qquad \forall t\in\{0,1,\dots,T-1\} \tag{10}$$

Note that inventory variables do not appear in the objective function or anywhere else in the model, hence using equation (9), we can substitute out these variables to obtain the following equivalent constraints:

$$\sum_{i\in I}\sum_{s=0}^{T-p_i}\sum_{l=0}^{t}\sum_{\substack{j\in\{1,\dots,h_i\}:\\ S_{is}^j=l}} \delta_i^j x_{is} \le I_{-1} + \sum_{l=0}^{t}\rho_l - K_t^{min} \quad \forall t\in\{0,1,\dots,T-1\} \tag{11}$$

$$\sum_{i\in I}\sum_{s=0}^{T-p_i}\sum_{l=0}^{t}\sum_{\substack{j\in\{1,\dots,h_i\}:\\ S_{is}^j=l}} \delta_i^j x_{is} \ge I_{-1} + \sum_{l=0}^{t}\rho_l - K_t^{max} \quad \forall t\in\{0,1,\dots,T-1\} \tag{12}$$

4.3. Polyhedral Study of Single Machine Scheduling with Sequence Dependent Setup Times

Even solving the downstream IP model on its own, presents challenges: the commercial solvers find it very hard to close the optimality gap. In this subsection, to alleviate this difficulty, we study the polyhedral properties of the time-indexed formulation for single machine scheduling with sequence dependent setup times problem as it constitutes an important basic component of the downstream scheduling model. Note that once we disregard the inventory (11, 12) and the capacity constraints

(5, 6, 8), our model becomes separable with respect to the casts assigned to different casters. So, for the purposes of the analysis in this section, its enough to consider the case where there is only one caster, i.e., $C = \{c\}$ and $I = J(c)$. From now on, we assume that $p_i \leq T \quad \forall i \in I$ as otherwise we can simply eliminate that job from the set I in the preprocessing phase. Let $I = \{1, 2, \ldots, n\}$ denote the set of jobs and $\{0, 1, \ldots, T-1\}$ denote the available time periods. Now our polyhedral set P is given by:

$$\sum_{t=0}^{T-p_i} x_{it} \leq 1 \qquad \forall i \in I \tag{13}$$

$$\sum_{i \in I} \sum_{s=\max\{t-p_i+1,0\}}^{\min\{t,T-p_i\}} x_{is} \leq 1 \qquad \forall t \in \{0, 1, \ldots, T-1\} \tag{14}$$

$$x_{it} + \sum_{s=\max\{t-p_j-R_{ji}+1,0\}}^{\min\{t+p_i+R_{ij}-1,T-p_j\}} x_{js} \leq 1 \qquad \forall t \in \{0, 1, \ldots, T-p_i\}, \quad \forall i, j \in I \tag{15}$$

$$x_{it} \in \{0, 1\} \qquad \forall t \in \{0, 1, \ldots, T-p_i\}, \quad \forall i \in I \tag{16}$$

In a feasible solution to our problem all of the jobs need not be scheduled therefore our problem is a variant of Traveling Salesman Problem (TSP), namely Orienteering problem. As Orienteering problem is NP-Hard, and our problem reduces to it, so is our problem. We refer to the schedules in which all of the jobs hasn't been scheduled as partial schedules.

van den Akker et al. (1999) studied the convex hull of (13) and (14) only. Here we extend their results to the sequence dependent setup case by studying the convex hull of (13)-(15).

The proofs of the following lemmas are omitted as they are easily shown by taking the appropriate set of unit vectors.

LEMMA 1. *(P) is full dimensional and $dim(P) = \sum_{i \in I}(T - p_i)$.*

LEMMA 2. *$x_{it} \geq 0$ is a facet of P for all $t \leq T - p_i$ and $i \in I$.*

A set $V \subseteq \{0, 1\}^n$ is called *down-monotone* if for all $x, y \in \{0, 1\}^n$ we have that $x \leq y$ and $y \in V$ implies that $x \in V$. Down-monotone 0-1 polytopes are polytopes that are the convex hull of a down-monotone subset of $\{0, 1\}^n$. Next we state the following lemma about down-monotone polytopes from Hammer et al. (1975).

LEMMA 3. *Let S be a down-monotone 0-1 polytope. A facet inducing inequality $a^T x \leq b$ for S with integral coefficients a_j and integral right hand side b has either $b > 0$ and coefficients $a_j \in \{0, 1, \ldots, b\}$ or it is a positive scalar multiple of $-x_j \leq 0$ for some j.*

It is easy to see that our polytope P is down-monotone whenever $R_{ik} \leq R_{ij} + p_j + R_{jk}$ for all distinct $i, j, k \in I$. From now on we assume that this relation holds and hence P is down-monotone. Note that this is a very realistic condition and in the real life instances we have, this condition is satisfied. Moreover, in the cases when this condition is violated the derived inequalities will still be valid however they may not induce facets of P.

Using Lemma 3, we have that all facet defining inequalities for P with right hand side 0 have the form $-x_{it} \leq 0$.

Following the notation in van den Akker et al. (1999), we introduce V as the index-set of variables with nonzero coefficients in an inequality. Let the set of variables with nonzero coefficients in an inequality associated with job i define a set of time periods $V_i = \{s : (i, s) \in V\}$. If job i is started in period $s \in V_i$, then we say that job i is started in V. With each set V_i we associate the parameters $L_i = \min\{s : s \in V_i\}$ and $U_i = \max\{s : s \in V_i\}$. For convenience, let $L_i = \infty$ and $U_i = -\infty$ when $V_i = \emptyset$. Clearly, if $V_i \neq \emptyset$, L_i (U_i) is the first (last) period that job i can be started in V.

We define an interval $[t_1, t_2]$ as the set of periods $\{t_1, t_1 + 1, \ldots, t_2\}$, i.e., the set of periods between time t_1 and time t_2. If $t_1 > t_2$, then $[t_1, t_2] = \emptyset$. For presentational convenience, we use $x(V)$ to denote $\sum_{(i,s)\in V} x_{is}$. Recall that P is a down-monotone 0-1 polytope. As a consequence of Lemma 3, valid inequalities with right hand side 1 will be denoted by $x(V) \leq 1$.

In the sequel of complete characterization of facet inducing inequalities with right hand side 1 for polytope P, we first show the structural properties of facet defining inequalities in van den Akker et al. (1999) are also valid in our case. These properties follow from the observation that a valid inequality $x(V) \leq 1$ can be facet inducing only if it is maximal, i.e., if there does not exist a valid inequality $x(W) \leq 1$ with $V \subset W$ where V is a proper subset of W. After these structural properties, we derive classes of inequalities that contain all facet inducing inequalities with right hand side 1. Finally, we show that the maximality is also sufficient as in van den Akker et al. (1999).

LEMMA 4. *A facet defining inequality $x(V) \leq 1$ for P is maximal.*

The following claim and its proof can be adapted directly from van den Akker et al. (1999).

CLAIM 1. *If $x(V) \leq 1$ is valid and maximal, then the sets V_i are intervals, i.e., $V_i = [L_i, U_i]$, for $i \in I$.*

The next lemma characterizes the intervals that are valid and maximal, we provide its proof in Appendix.

LEMMA 5. *Let $x(V) \leq 1$ be valid and maximal. Then the sets V_i satisfy the following relations:*

$$V_i = [L_i, \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1] \quad \textit{and} \quad V_i = [\max_{k \neq i}\{U_k - R_{ik}\} - p_i + 1, U_i].$$

Consequently, Lemma 4 and Lemma 5 can be combined to give the following theorem.

THEOREM 1. *A facet inducing inequality $x(V) \leq 1$ for P has the following structure:*

$$V_i = [L_i, \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1] = [\max_{k \neq i}\{U_k - R_{ik}\} - p_i + 1, U_i]. \tag{17}$$

Note that when $R_{ij} = 0$ for all $i, j \in I$, our Theorem 1 reduces to Theorem 1 of van den Akker et al. (1999). Clearly only maximal valid inequalities will be facet defining, hence we have:

COROLLARY 1. *A valid inequality $x(V) \leq 1$ with structure* (17) *that is maximal is facet inducing for P.*

Note that an inequality is maximal if and only if it is impossible to add more variables to it. In order to give explicit descriptions of facets with right hand side 1, we need to derive conditions for which valid inequality $x(V) \leq 1$ with structure (17) is maximal. The cases in which inequalities with structure (17) are not maximal can be grouped into two:

- The first case is if $V_j = \emptyset$ for all $j \neq i$ and not all variables belonging to job i are included. Clearly, now the missing variables belonging to job i can be added. This case is excluded by the condition that either $V_j \neq \emptyset$ for some $j \neq i$ or $L_i = 0$ and $U_i = T - p_i$. In addition to this if $L_i = 0$, $U_i = T - p_i$ for some $i \in I$, and $V_k = \emptyset$ for all $k \in I \setminus \{i\}$, then the inequalities with structure (17) coincide with the inequalities (13).
- The second case is if the inequality is at the border of the interval [0,T-1] and some variables included in the structure (17) are missing because they are outside the domain $\{x_{it} : \ i \in I, t \in \{0, 1, \ldots, T - p_i\}\}$. Now, it may be possible to add variables outside the structure.

4.3.1. Strengthening Downstream Formulation In the rest of this section, we provide some strengthened form of the original constraints in our IP formulation and then to tighten its LP-relaxation, we introduce some valid inequalities based on facets with right hand side 1 derived in the previous section.

Note that when we have $U_i = t \ \ \forall i \in J(c)$, using the structure of (17), we arrive at the following inequality (18):

$$\sum_{i \in J(c)} \sum_{s=\max\{t-p_i-\min\limits_{k \in J(c)\setminus i}\{R_{ik}\}+1,0\}}^{\min\{t, T-p_i\}} x_{is} \leq 1 \quad \forall t \in \{0, 1, \ldots, T-1\}, \quad \forall c \in C. \tag{18}$$

Clearly (18) is a lifted form of the caster capacity constraints (3). Moreover, when we have $\max_{i\in J(c)}\{p_i + \min_{k\in J(c)\setminus i}\{R_{ik}\}\} - 1 \le t \le T - \max_{i\in J(c)}\{p_i\}$, these inequalities are maximal and hence facet defining for the respective single machine scheduling with sequence dependent setup times polytope, P.

Suppose $U_q = t + v$ and $U_i = t \quad \forall i \in J(c_q)\setminus q$ for some $v \ge 0$. Then using (17), we get $L_q = t - p_q - \min_{k\in J(c_q)\setminus q}\{R_{qk}\} + 1$, and $L_i = t - p_i - \min\{R_{iq} - v, \min_{k\in J(c_q)\setminus\{i,q\}}\{R_{ik}\}\} + 1$ and obtain the following set of inequalities:

$$\sum_{i\in J(c_q)\setminus q}\sum_{s=\max\{t-p_i-\min\{R_{iq}-v,\min_{k\in J(cq)\setminus\{i,q\}}\{R_{ik}\}\}+1,0\}}^{\min\{t,T-p_i\}} x_{is} + \sum_{s=\max\{t-p_q-\min_{k\in J(cq)\setminus q}\{R_{qk}\}+1,0\}}^{\min\{t+v,T-p_q\}} x_{qs} \le 1. \quad (19)$$

It is easy to see that inequalities (19) are valid for our polytope P. Furthermore, inequalities (19) are facet defining for P whenever they are maximal. Note that when $v \ge \nu_q^*$ where $\nu_q^* = \max_{i\in J(c_q)\setminus q}\left\{p_i + \min_{k\in J(c_q)\setminus i}\{R_{ik}\}\right\}$, then inequality will have $V_i = \emptyset \quad \forall \in J(c_q)\setminus q$, and depending on the value of v, we might have $V_q \subseteq [0, T - p_q]$ and hence (19) wouldn't be maximal and the corresponding inequality will be dominated by constraint (2) included in the original formulation.

Also when we don't have setup times, i.e., $R_{ij} = 0\ \forall i, j \in I$, our inequality reduces to the inequalities given in van den Akker et al. (1999):

$$\sum_{i\in J(c_q)\setminus q}\sum_{s=\max\{t-p_i+v+1,0\}}^{\min\{t,T-p_i\}} x_{is} + \sum_{s=\max\{t-p_q+1,0\}}^{\min\{t+v,T-p_q\}} x_{qs} \le 1$$

For all $0 \le v < \nu_q^*$ where $\nu_q^* = \max\{p_i :\ i \in J(c_q)\setminus q\}$, these inequalities are proved to contain all the facets with right hand side 1 of the corresponding polytope without setup times.

By taking $L_i = t \quad \forall i \in J(c)$, using the structure of (17), we can obtain another set of valid inequalities:

$$\sum_{i\in J(c)}\sum_{s=t}^{\min\{t+\min_{k\in J(c)\setminus i}\{p_k+R_{ki}\}-1,T-p_i\}} x_{is} \le 1 \quad \forall t \in \{0,1,\ldots,T-1\}, \quad \forall c \in C. \quad (20)$$

Suppose $L_q = t - v$ and $L_i = t \quad \forall i \in J(c_q)\setminus q$ for some $v \ge 0$. Then using (17), we get $U_q = t + \min_{k\in J(c_q)\setminus q}\{p_k + R_{kq}\} - 1$, and $U_i = t + \min\{p_q + R_{qi} - v, \min_{k\in J(c_q)\setminus\{i,q\}}\{p_k + R_{ki}\}\} - 1$ and obtain the following set of inequalities:

$$\sum_{i\in J(c)\setminus q}\sum_{s=t}^{\min\{t+\min\{p_q+R_{qi}-v,\min_{k\in J(cq)\setminus\{i,q\}}\{p_k+R_{ki}\}\}-1,T-p_i\}} x_{is} + \sum_{s=\max\{t-v,0\}}^{\min\{t+\min_{k\in J(cq)\setminus q}\{p_k+R_{kq}\}-1,T-p_q\}} x_{qs} \le 1. \quad (21)$$

It is easy to see that inequalities (21) are valid for our polytope P. Furthermore, inequalities (21) are facet defining for P whenever they are maximal. Note that when $v \ge \mu_q^*$ where $\mu_q^* =$

$\max_{i \in J(c_q) \setminus q} \left\{ \min_{k \in J(c_q) \setminus i} \{p_k + R_{ki}\} \right\}$, then inequality will have $V_i = \emptyset \quad \forall \ \in J(c_q) \setminus q$, and depending on the values of v and t, we might have $V_q \subseteq [0, T - p_q]$ and hence (21) wouldn't be maximal and the corresponding inequality will be dominated by constraint (2) included in the original formulation.

Finally a simple valid inequality derived from the basic idea of packing jobs to machines and considering their processing time requirements is as follows:

$$\sum_{i \in J(c)} \sum_{t=0}^{T-p_i} p_i x_{it} \leq T \quad \forall c \in C \tag{22}$$

5. Logic-based Benders Decomposition

The main idea behind logic-based Benders decomposition is to model the problem in different stages and then use the information in the latter stages to generate valid inequalities for the earlier stage. In our case, we have only two stages. Our first stage is the downstream scheduling where we solve a time-indexed IP formulation to obtain the starting time of each cast on the caster. Let $U(\bar{x})$ denote the set of upstream schedules which are feasible with respect to the downstream solution $\bar{x}$. Given the start times of the casts, i.e., $\bar{x}$, if the upstream problem is feasible, then we obtain a feasible solution for our overall problem. Whenever the upstream problem is infeasible, we know that any solution $\tilde{x}$ which selects a superset of casts containing all the casts in $\bar{x}$ and assigns the same start times as in $\bar{x}$, i.e., $\tilde{x}_{it} = \bar{x}_{it} \quad \forall (i,t)$ such that $\bar{x}_{it} = 1$, will also lead to an infeasible upstream schedule, i.e., $U(\tilde{x}) = \emptyset$. Therefore, in this case, the following inequality is a valid Bender's cut which eliminates all such solutions $\tilde{x}$

$$\sum_{(i,t) \text{ s.t. } \bar{x}_{it}=1} (1 - x_{it}) \geq 1.$$

The overall performance of Benders decomposition depends heavily on the amount of infeasible region eliminated by the Benders cuts developed throughout the algorithm hence it's customary to strengthen the Benders cuts and add more than one cut at each iteration.

Note that as we shift the start time of each cast by the same amount, the resulting schedule will still lead to an infeasible upstream schedule as the resource capacities do not vary over time. Thus, whenever we identify a solution $\bar{x}$ such that $U(\bar{x}) = \emptyset$, we can add the following group of Benders cuts:

$$\sum_{(i,t) \text{ s.t. } \bar{x}_{it}=1} (1 - x_{it+\Delta}) \geq 1 \qquad \forall -t^{min}_{\bar{x}} \ \leq \ \Delta \ \leq \ t^{max}_{\bar{x}} \tag{23}$$

where $t^{min}_{\bar{x}} = \min\{t : \ i \text{ s.t. } \bar{x}_{it} = 1\}$ and $t^{max}_{\bar{x}} = T - 1 - \max\{t + p_i : \ i \text{ s.t. } \bar{x}_{it} = 1\}$.

Clearly solving a single B&B tree by accumulating and using all of the Benders cuts developed in all open nodes leads to performance increases in terms of computational time. Moreover, from

Algorithm 1 Branch-and-Check

Require: A B&B solver for the time-indexed formulation with x_{it} variables and an algorithm to check the upstream feasibility of any given solution from the time-indexed formulation x_{it}.

Set $0 \leftarrow x^*$, $0 \leftarrow z^*$, $k = 0$.

Solve the LP-relaxation of the master problem.

while There are open B&B nodes **do**

Continue the B&B tree until an integer feasible solution x^k for the master problem is found.

Fix x^k and solve $U(x^k)$ using constraint programming.

if $U(x^k) = \emptyset$ **then**

Reject x^k as incumbent.

Add cuts $\sum_{(i,t):x^k_{it}=1} (1 - x_{it+\Delta}) \geq 1 \quad \forall -t^{min}_{x^k} \leq \Delta \leq t^{max}_{x^k}$ to the master problem at all open B&B nodes.

Set $k + 1 \leftarrow k$.

end if

if $U(x^k) \neq \emptyset$ **then**

Compute $z(x^k)$. If $z(x^k) \geq z^*$, then update incumbent in B&B tree by setting $x^k \leftarrow x^*$, $z(x^k) \leftarrow z^*$.

end if

end while

our experiments with the time-indexed formulations, we observe that even LP-relaxation solution times can be in the order of minutes for our instances. So our implementation of logic-based Benders decomposition relies on the idea of *Branch-and-Check* presented in Thorsteinsson (2001). Algorithm 1 gives the pseudo code of our implementation.

6. Impact of the Scheduler

The scheduling system described in this paper was just one part of a large, multi-million dollar development project that interacted with an upstream optimization module to design the casts that are input to the system, and a downstream module to perform hot strip mill sequencing of the production scheduled by the system. The scheduling system module was developed over a period of 18 months. The users acknowledge that this scheduling problem is extremely complex, in part due to the complexity of the manufacturing processes and the wide diversity of product types produced. There were several attempts in the past by the user to develop a scheduling system in-house. This included developing a rule-based system and experimenting with heuristic scheduling

approaches, but the results were unsatisfactory. As of writing, the system has been deployed and feedback has been very favorable. The end users are very impressed that the IBM system is able to generate schedules that dominate hand generated schedules. Furthermore, the system can generate full 2-day schedules within 5-10 minutes (using a scheduling horizon of 8 hours) on a 3GHz Intel Xenon machine (running RedHat Enterprise Linux 4.5) as opposed to many hours needed for the human experts to generate a schedule. This is important, since in practice, the users frequently use the system to perform some kind of what-if analysis, experimenting with problem parameters and upstream optimization modules to generate and select from multiple possible schedules. Some real-time adjustment of the schedules generated by the system is performed by the users during execution due to the quite dynamic nature of the shop floor.

Once we had a scheduling system that was able to generate feasible schedules accommodating all the complex constraints from the floor shop the next aspect of the work was focused on tuning the scheduler to handle different objectives. The main goal was to provide the users a quick way to evaluate tradeoff between throughput (measured in terms of number of charges scheduled in a given scheduling horizon) vs the number of early PST slabs (number of slabs that meet desired start times). To handle these considerations, we provided two different objectives for the scheduler:

(i) Maximize the number of charges scheduled, and

(ii) Maximize the total score where the score for each cast is based on number of slabs that meet desired PST.

As a result of the request of plant operators on limiting the tradeoff to ensure that productivity is not impacted, the final deployment of the engine used the second objective and the throughput was specified as a constraint. The limits on throughput were set based on current throughput levels achieved by the operators using manual scheduling. In determination of the limits, several data sets which are representative of different operational modes characterized by maintenance schedules and hot metal metal inventory levels, were used.

6.1. Quantifying the Impact

A key performance metric for the scheduling engine is how the model-optimal casting times compare to the desired planned start times (PST). Recall that the cast's PST is set using a high-level planning engine which doesn't have full visibility into shop floor operations and usually results in PSTs that might not be feasible with respect to the shop floor. We provide a comparison of the results achieved using our scheduling engine with the desired PSTs in Figure 4 below. This figure provide a histogram of the number of slabs that were scheduled within a time bucket defined as the difference between the desired PST and actual scheduled time (by scheduling engine). This

histogram is based on over 35 data sets for which the engine generated a schedule for 6 shifts each. Note that a zero on the x-axis indicates the slabs that were scheduled on the desired PST. Positive numbers on the x-axis is the delay in scheduling (in days) as compared to the desired PST. Negative numbers provide a count of the overdue slabs(PSTs before the schedule start date) that were scheduled within he scheduling horizon. In general it is very desirable to schedule overdue slabs, and to schedule slabs within a day of their PST. We find that there is a sizable tail of slabs with PSTs into the future that get scheduled with the scheduling horizon. The batching requirements of charges into casts lead to the grouping of early PST slabs with late PST slabs and thereby force the choice of future orders in turn denying the early orders a chance to be scheduled.

Using the scheduling engine, over 39% of casts are scheduled on the desired PST. Over 71% of casts are scheduled within plus or minus one day of the prescribed PST date. The remaining 29% is accounted by orders that are built ahead of their PSTs (due to batching constraints). Our engine's results were considered to be excellent by the production planners and helped to boost the company's on-time delivery rate from 75% to over 90% while keeping productivity at the same and/or improved level.

6.2. Sample Computational Results

In this section, we provide details related with the effectiveness of our strengthening and valid inequalities on the problem formulation as well as the logic based Benders approach. The main properties of 3 real life data instances used in these experiments are summarized in Table 2.

We transform the processing and setup times given in Table 2 into discrete time periods with a length of 15 minutes by dividing them with 15 and rounding up when necessary in our time-indexed formulation. To be consistent in our models, we choose to underestimate the BOF capacity usage.

The code is written in c++ and the IP model is built and solved with Concert 2.5 in ILOG Cplex 11.1. All of the experiments are done on an IBM Intellistation ZPro machine with dual Intel 3GHz Xeon CPUs each with two processor cores, 16GB RAM running on a RedHat Enterprise Linux platform.

We set the LP algorithm at the root node to be primal simplex as it takes around 30-70 seconds for $T = 16$ problems, whereas dual simplex takes about 400-800 seconds, and barrier takes about 300-3000 seconds. The reoptimization of LPs is faster with dual simplex, so we set the node algorithm to dual simplex. The presolve operations, especially after finding a good feasible solution are quite powerful as they result in fixing the values of some of the variables, thus reducing the size of the model hence we set the "RepeatPresolve" parameter in Cplex to its highest value. Also, from our initial experiments with time-indexed formulations, we observe that finding good initial

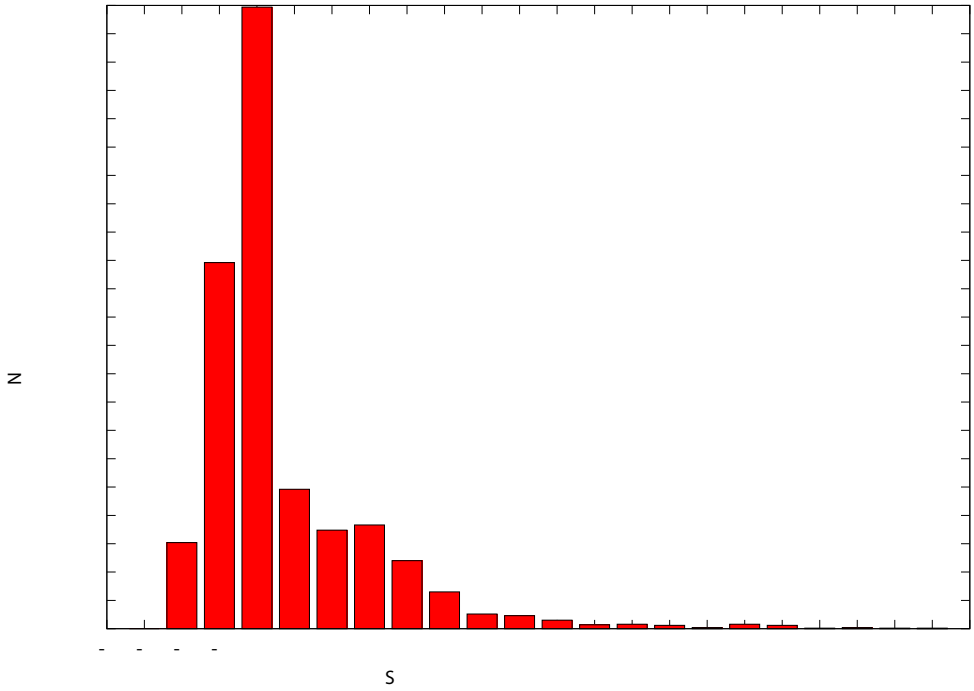

Figure 4 **Histogram of Difference Between Customer's Preferred Casting Time and Our Engine's Prescribed Casting Time.**

solutions have a big impact on making the solution process faster. Therefore, in all of the following experiments, we set the search emphasis of Cplex to "hidden feasible solutions". In addition, clique, cover, GUB cover and Gomory fractional cuts turn out to be most frequently generated ones by Cplex, hence we set their generation frequency to the highest. We put a time limit of 3600 seconds on each instance for time horizons of both $T = 8$ hours and $T = 16$ hours.

We report our computational results in Tables 3 and 4 for the time horizon of $T = 8$ hours and $T = 16$ hours respectively. The columns **(Basic)** refer to the original formulation without the strengthening of the original constraints or any valid inequalities we provide in section 4.3.1. The columns **(Strengthened)** refer to the formulation where we lift the original constraints as given in (18) and add the valid inequalities (20) and (22) as constraints to the model. In the **(Strengthened)** formulation, we also separate the valid inequalities (19) and (21) at each fractional node in the $B\&B$ tree.

For each instance, we give the objective value of the optimal (or best known) integer solution as **Best LB**. We also report the objective value for the initial LP relaxation before any of the standard cutting plane method is applied as **LP relax UB** and the corresponding **LP optimality gap(%)**

Table 2 Characteristics of the instances

	Problem A	*Problem B*	*Problem C*
# of casts, $\|I\|$:	93	106	60
# of casters, $\|C\|$:	6	6	6
# of casts assigned to each caster, $\{\|J(c)\| : c \in C\}$:			
min	5	8	5
max	39	36	22
avg	15.50	17.67	10.00
cast scores, $\{s_i : i \in I\}$:			
min	51	54	57
max	1350	1403.6	1403.6
avg	282.81	831.26	838.30
# charges of each cast, $\{h_i : i \in I\}$:			
min	1	1	1
max	8	8	8
avg	4.17	4.08	3.47
processing time of casts (in terms of minutes), $\{p_i : i \in I\}$:			
min	31	26	30
max	390	408	378
avg	207.00	206.87	191.78
setup time between casts (in terms of minutes), $\{R_{ij} : i, j \in J(c), c \in C\}$:			
min	0	0	0
max	140	180	120
avg	99.19	99.69	87.59
$\|\{(i,j) : R_{ij} > 0\}\|$:	2399	2698	884

which is computed as $\frac{\text{(LP relax UB - best LB)}}{\text{|best LB|}} \times 100$. Whenever Cplex stops due to the time limit, we report the final value of the upper bound as **Best UB**, where relevant we also provide **Final optimality gap (%)** which is computed as $\frac{\text{(Best UB - best LB)}}{\text{|best LB|}} \times 100$. We also report the solution time in seconds, the number of Branch and Bound nodes processed, the total number of simplex iterations and the total number of user cuts (19) and (21) separated. We provide some statistics related with the logic-based Benders decomposition in the last two rows of these tables. Recall that whenever we have a feasible incumbent solution in the $B\&B$ process, we use CP techniques to check for its feasibility in the upstream processes, we provide the number of times we encountered a feasible solution in **# calls to upstream scheduler**. If IP solution turns out to be infeasible, then we generate and add the corresponding benders cuts given in (23) to our problem. This number is reported as **# logic based benders cuts**.

As can be seen from the results in Table 3, the strengthening has a big impact in terms of reducing the initial LP gap. Except the first data instance, the strengthened model together with the valid inequalities performs better than the basic formulation in terms of solution time, $B\&B$ nodes and simplex iteration counts. This performance is quite remarkable for the last instance, *Problem C*, where the basic model cannot be solved within the time limit of an hour however the strengthened one is solved within 2252 seconds. Note that as the number of cuts separated increases among the instances, the performance of strengthened formulation becomes more pronounced. This is a nice

Table 3 Performance of time-indexed formulations with $T = 8$ hours

	Problem A		*Problem B*		*Problem C*	
	(Basic)	**(Strengthened)**	**(Basic)**	**(Strengthened)**	**(Basic)**	**(Strengthened)**
Best LB (Optimal)	10553.30	10553.30	17956.00	17956.00	14095.40	14095.40
LP relax UB	12088.24	10992.17	20437.41	19093.36	16866.43	15976.38
LP optimality gap (%)	14.54	4.16	13.82	6.33	19.66	13.34
Best UB	-	-	-	-	14268.50	-
Solution time (seconds)	85.70	252.61	389.78	309.91	3600.01*	2251.65
# $B\&B$ nodes	285	3881	32430	15939	37645	22382
# simplex iterations	18235	135047	907670	505666	3338143	2047727
# user cuts separated	-	75	-	266	-	571
# calls to upstream scheduler	10	6	9	5	7	7
# logic based benders cuts	0	0	0	0	0	0

indicator showing the strength of the inequalities developed in section 4.3.1.

Table 4 Performance of time-indexed formulations with $T = 16$ hours

	Problem A		*Problem B*		*Problem C*	
	(Basic)	**(Strengthened)**	**(Basic)**	**(Strengthened)**	**(Basic)**	**(Strengthened)**
Best LB	16917.30	16917.30	32915.20	32915.20	22663.29	22663.29
LP relax UB	18235.09	17347.12	35623.60	34303.29	28639.14	24944.45
LP optimality gap (%)	7.79	2.54	8.23	4.22	26.37	10.07
Best UB	17143.96	17140.70	33880.06	33846.96	24299.99	24158.44
Final optimality gap (%)	1.34	1.32	2.93	2.83	7.22	6.60
# $B\&B$ nodes	76	52	30	2	41	7
# simplex iterations	119532	104714	274653	156757	418013	327341
# user cuts separated	-	754	-	1494	-	861
# calls to upstream scheduler	8	3	5	3	3	2
# logic based benders cuts	0	0	0	0	0	0

Similar to the previous results, the improvement of the initial LP gaps obtained by strengthening is clear in Table 4. Note that the number of variables and constraints in the formulation increases significantly when we double the time horizon to $T = 16$ hours. Due to this increase, the solution times for reoptimization of the LP relaxations become significant. Although, within a time limit of 3600 seconds, we cannot prove to close the optimality gap in any of the instances, we provide some details related with the solution progress. We observe that final gaps are relatively small however, it is almost impossible to close those gaps by $B\&B$ as the number of $B\&B$ nodes that are processed within the time limit is well below 100. We also observe that the strengthened formulation can process far fewer nodes within the same time limit. This is mainly due to the cut separation process since after each phase of cut separation LP is reoptimized and we look for other violated inequalities. This procedure continues until no violated cuts of form (19) or (21) are found. On the other hand, the time spent for separating these cuts is well deserved as the strengthened model together with the valid inequalities performs better than the basic formulation in terms of the final optimality gap for all instances.

A final note is on the effectiveness of the BOF capacity profile estimation. As can be seen from the above tables, none of the incumbent solutions found throughout the $B\&B$ process turns out to be infeasible in terms of the upstream processes. This demonstrates our effective integration of IP and CP models through BOF capacity estimation.

7. Conclusions and Further Remarks

As manufacturing companies turn their attention from planning to incorporating their plans into operations, optimization-based operations scheduling software plays a key role in achieving this. Such a need can hardly be more critical than that of the steel industry as they face ever increasing demand, high capital intensive manufacturing equipments and seek to increase profits by both increasing their productivity and customer satisfaction at the same time.

This paper describes the use of an integrated IP-CP approach to improve scheduling operations in the hot metal shop of a steel plant. Although we have presented our models in this settings, the techniques developed here are applicable to problems with time-critical flowshop structure. Specifically in the steel setting, we model the operations related with downstream resources (casters) through time-indexed IP formulation and the upstream resources (BOF, RF) through CP techniques. Furthermore, we observe the importance of incorporating upstream capacity requirements while planning for downstream scheduling. The foundation of our approach lies in the fact that once the cast start times on the casters are given, we can easily estimate the BOF capacity requirements of each cast. In addition to this, the definition of time-indexed variables allows us to represent these estimated BOF capacity requirements without introducing additional variables, hence we choose to model the upstream process using these estimated capacity inequalities.

The time-indexed formulations in general provide tight bounds from their LP-relaxations at the expense of huge number of variables and constraints required in the formulation. Although the initial LP gaps are small, we observed that the large scale of the problem leads to very large number of variables, which makes it almost impossible to close even the smallest gaps. Therefore we further enhance our time-indexed IP formulation with additional valid inequalities derived from the single machine scheduling with sequence dependent setup times polytope. The effectiveness of our approach is presented with the results obtained from real life instances.

Finally, the time spent for solving linear relaxation of time-indexed formulations tends to increase drastically with increasing time horizon. In larger instances, Barrier and Dual simplex algorithms end up spending times in terms of tens of minutes just to solve the initial LP-relaxations. Although the Primal simplex algorithm performs better still the LP solution times remains in terms of

minutes instead of seconds. As B&B based approaches depend on LP-relaxations and frequent re-optimization of LPs, it is important to improve LP solution times. There have been successful research on implementing column generation techniques for time-indexed formulations, so adopting their approaches to our setting and developing a framework which combines together Branch-and-Price and Branch-and-Check will be quite interesting and most probably successful.

Appendix. Proof of Lemma 5

L et $i \in I$ and assume that $V_i \neq \emptyset$, then by Claim 1, we have $V_i = [L_i, U_i]$.

We will first show that $U_i = \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1$.

Observe that $x_{kL_k} = 1$ and $x_{it} = 1$ defines a feasible schedule for any $t \geq L_k + p_k + R_{ki}$ and for all $k \in I \setminus \{i\}$. Since $x(V) \leq 1$ is a valid inequality, only one of the pairs (k, L_k) or $(i, L_k + p_k + R_{ki})$ can occur in V. By the definition of L_k, we have $(k, L_k) \in V$ which implies that $U_i \leq L_k + p_k + R_{ki} - 1$ for all $k \in I \setminus \{i\}$. Thus we have $U_i \leq \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1$.

Now let $x_{it} = 1$ for some $t \in [L_i, \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1]$. Let's assume for contradiction that there exists a job $j \neq i$ with $(j, s) \in V$ such that $x_{it} = x_{js} = 1$ defines a feasible schedule. If $t < s$, i.e., job j is started after job i, then by shifting the start of job i to L_i we will still have a feasible schedule. Clearly, (i, L_i) and $(j, s) \in V$ however $x_{iL_i} = x_{js} = 1$ violates $x(V) \leq 1$, which is a contradiction. So no job in V can be processed after job i. Now if $t > s$, i.e., job j is started before job i, then the earliest time job j can be finished will be $L_j + p_j$. Hence the earliest time job i can be started after job j will be $L_j + p_j + R_{ji}$, so $s \geq L_j + p_j + R_{ji}$. But this is a contradiction since $s \leq U_i \leq \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1 < L_j + p_j + R_{ji}$. So no job in V can be processed before job i. Thus having $x_{it} = 1$ for some $t \in [L_i, \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1]$ eliminates the processing of all other jobs in V. The maximality of the inequality $x(V) \leq 1$ implies that $U_i = \min_{k \neq i}\{L_k + p_k + R_{ki}\} - 1$.

To finish the proof we will next show that $L_i = \max_{k \neq i}\{U_k - R_{ik}\} - p_i + 1$ by proceeding as we did in the previous case.

Observe that $x_{kU_k} = 1$ and $x_{it} = 1$ defines a feasible schedule for any $t \leq U_k - p_i - R_{ik}$ and for all $k \in I \setminus \{i\}$. Since $x(V) \leq 1$ is a valid inequality, only one of the pairs (k, U_k) or $(i, U_k - p_i + R_{ik})$ can occur in V. By the definition of U_k, we have $(k, U_k) \in V$ which implies that $L_i \geq U_k - p_i - R_{ik} + 1$ for all $k \in I \setminus \{i\}$. Thus we have $L_i \geq \max_{k \neq i}\{U_k - R_{ik}\} - p_i + 1$.

Now let $x_{it} = 1$ for some $t \in [\max_{k \neq i}\{U_k - R_{ik}\} - p_i + 1, U_i]$. Let's assume for contradiction that there exists a job $j \neq i$ with $(j, s) \in V$ such that $x_{it} = x_{js} = 1$ defines a feasible schedule. If $t > s$, i.e., job j is started before job i, then by shifting the start of job i to U_i we will still have a feasible schedule. Clearly, (i, U_i) and $(j, s) \in V$ however $x_{iU_i} = x_{js} = 1$ violates $x(V) \leq 1$, which is a contradiction. So no job in V can be processed before job i.

Now if $t < s$, i.e., job j is started after job i, then the earliest time job i can be finished will be $L_i + p_i$. Hence the earliest time job j can be started after job i will be $L_i + p_i + R_{ij}$, so

$$s \geq \underbrace{L_i}_{\geq \max_{k\neq i}\{U_k - R_{ik}\} - p_i + 1} + p_i + R_{ij} \;\geq\; \max_{k\neq i}\{U_k - R_{ik}\} + R_{ij} + 1 \;\geq\; U_j - R_{ij} + R_{ij} + 1.$$

But this is a contradiction since we assume that $(j, s) \in V$ i.e., using the definition of U_j, $s \leq U_j$. So no job in V can be processed after job i. Thus having $x_{it} = 1$ for some $t \in [\max_{k\neq i}\{U_k - R_{ik}\} - p_i + 1, U_i]$ eliminates the processing of all other jobs in V. The maximality of the inequality $x(V) \leq 1$ implies that $L_i = \max_{k\neq i}\{U_k - R_{ik}\} - p_i + 1$.

References

Cambazard, H., P.-E. Hladik, A.-M. Déplanche, N. Jussien, Y. Trinquet. 2004. Decomposition and learning for a hard real time task allocation problem. M. Wallace, ed., *CP*, *Lecture Notes in Computer Science*, vol. 3258. Springer, 153–167.

Chu, Y., Q. Xia. 2004. Generating benders cuts for a general class of integer programming problems. Régin and Rueher (2004), 127–141.

Chu, Y., Q. Xia. 2005. A hybrid algorithm for a class of resource constrained scheduling problems. R. Barták, M. Milano, eds., *CPAIOR*, *Lecture Notes in Computer Science*, vol. 3524. Springer, 110–124.

Corréa, A.I., A. Langevin, L.-M. Rousseau. 2004. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. Régin and Rueher (2004), 370–379.

Cowling, P., W. Rezig. 2000. Integration of continuous caster and hot strip mill planning for steel production. *Journal of Scheduling* **3** 185–208.

Crama, Y., F.C.R. Spieksma. 1996. Scheduling jobs of equal length: Complexity, facets and computational results. *Mathematical Programming* **72** 207–227.

Dash, S., J.R. Kalagnanam, C. Reddy, S.H. Song. 2007. Production design for plate products in the steel industry. *IBM Journal of Research and Development* **51** 345–362.

Davenport, A.J., J.R. Kalagnanam, C. Reddy, S. Siegel, J. Hou. 2007. An application of constraint programming to generating detailed operations schedules for steel manufacturing. C. Bessiere, ed., *CP*, *Lecture Notes in Computer Science*, vol. 4741. Springer, 64–76.

Dorn, J. 1996. Guest editor's introduction: Expert systems in the steel industry. *IEEE Expert* **11**(1) 18–21.

Dorn, J., R. Shams. 1996. Scheduling high-grade steelmaking. *IEEE Expert* **11**(1) 28–35.

Eremin, A., M. Wallace. 2001. Hybrid benders decomposition algorithms in constraint logic programming. Walsh (2001), 1–15.

Hammer, P.L., E.L. Johnson, U.N. Peled. 1975. Facets of regular 0-1 polytopes. *Mathematical Programming* **8** 179–206.

Hardin, J.R., G.L. Nemhauser, M.W.P. Savelsbergh. 2008. Strong valid inequalities for the resource constrained scheduling problem with uniform resource requirements. *Discrete Optimization* **5** 19–35.

Harjunkoski, I., I.E. Grossmann. 2001. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering* **25**(11-12) 1647–1660.

Harjunkoski, I., I.E. Grossmann. 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering* **26**(11) 1533–1552.

Hooker, J.N. 2000. *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons, New York, NY.

Hooker, J.N. 2006. An integrated method for planning and scheduling to minimize tardiness. *Constraints* **11**(2-3) 139–157.

Hooker, J.N., G. Ottosson. 2003. Logic-based benders decomposition. *Mathematical Programming* **96** 33–60.

Hooker, J.N., H. Yan. 1995. Logic circuit verification by benders decomposition. *Principles and Practice of Constraint Programming: The Newport Papers* 267–288.

Jain, V., I.E. Grossmann. 2001. Algorithms for hybrid milp/clp models for a class of optimization problems. *INFORMS Journal on Computing* **13** 258–276.

Kanet, J.J., S.L. Ahire, M.F. Gorman. 2004. Constraint programming for scheduling. J. YT. Leung, ed., *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, vol. 47. Chapman and Hall/CRC Press, Boca Raton, FL, 1–22.

Kumar, V., S. Kumar, M.K. Tiwari, F.T.S. Chan. 2006. Auction-based approach to resolve the scheduling problem in the steel making process. *International Journal of Production Research* **44**(8) 1503–1522.

Lally, B., L. Biegler, A. Henein. 1987. A model for sequencing a continuous casting operation to minimize costs. *Iron & Steel maker* **10** 53–70.

Lawler, E.L., J.K. Lenstra, A.H.G.R. Kan, D.B. Shmoys. 1993. Sequencing and Scheduling: Algorithms and Complexity. S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin, eds., *Handbooks of Operations Research and Management Science:Logistics of Production and Inventory*, vol. 4. Elsevier Science, North Holland, 445–522.

Lee, H.S., S.S. Murthy, S.W. Haider, D.V. Morse. 1996. Primary production scheduling at steelmaking industries. *IBM Journal of Research and Development* **40**(2) 231–252.

Maravelias, C.T., I.E. Grossmann. 2004. Using milp and cp for the scheduling of batch chemical processes. Régin and Rueher (2004), 1–20.

Missbauer, H., W. Hauber, W. Stadler. 2008. A scheduling system for the steelmaking-continuous casting process. a case study from the steel-making industry. *International Journal of Production Research* DOI: 10.1080/00207540801950136.

Pan, Y., L. Shi. 2007. On the equivalence of the max-min transportation lower bound and the time-indexed lower bound for single-machine scheduling problems. *Mathematical Programming* **110**(3) 543–559.

Queyranne, M., A.S. Schulz. 1994. Polyhedral approaches to machine scheduling. Preprint 408, Department of Mathematics, Technische Universitat, Berlin. Revisited in June 1997.

Régin, J.-C., M. Rueher, eds. 2004. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004, Proceedings*, *Lecture Notes in Computer Science*, vol. 3011. Springer.

Sadykov, R. 2008. A Branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research* **189**(3) 1284–1304.

Sousa, J.P., L.A. Wolsey. 1992. A time-indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming* **54** 353–367.

Tang, L., J. Liu, A. Rong, Z. Yang. 2000. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research* **120**(2) 423–435.

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

Tang, L., P.B. Luh, J. Liu, L. Fang. 2002. Steel-making process scheduling using lagrangian relaxation. *International Journal of Production Research* **40**(1) 55–70.

Tang, L., G. Wang, J. Liu. 2007. A branch-and-price algorithm to solve the molten iron allocation problem in iron and steel industry. *Computers and Operations Research* **34**(10) 3001–3015.

Thorsteinsson, E.S. 2001. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. Walsh (2001), 16–30.

Timpe, C. 2002. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum* **24**(4) 431–448.

van den Akker, M., C.A.J. Hurkens, M.W.P. Savelsbergh. 2000. Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing* **12**(2) 111–124.

van den Akker, M., S.P.M. van Hoesel, M.W.P. Savelsbergh. 1999. A polyhedral approach to single-machine scheduling problems. *Mathematical Programming* **85** 541–572.

Walsh, T., ed. 2001. *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings*, *Lecture Notes in Computer Science*, vol. 2239. Springer.

Waterer, H., E.L. Johnson, P. Nobili, M.W.P. Savelsbergh. 2002. The relation of time indexed formulations of single machine scheduling problems to the node packing problem. *Mathematical Programming* **93**(3) 477–494.