

# Scheduling Steel Production using Mixed-Integer Programming and Constraint Programming

Andrew Davenport, Jayant Kalagnanam

IBM T.J.Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598, USA,

{davenport, jayant}@us.ibm.com

---

We present an overview of a system we have developed for generating short-term production schedules for a large steel manufacturer. We have developed a decomposition based approach which uses mixed-integer programming to generate a plan for production on the downstream casting processes, and constraint programming to generate a schedule upstream of these casting processes, taking into account detailed scheduling constraints and preferences. One drawback to this approach is that downstream production plans can be generated which cannot be scheduled on the upstream processes, since downstream planning does not take into account bottlenecks that may occur when scheduling upstream. We investigate an integration mechanism, which uses constraint programming to build local estimations of upstream capacity utilization, which are used in the integer programming formulation for downstream production planning. We present results illustrating the effectiveness of this approach.

*Keywords:* Applications, Production Scheduling

---

## 1. Introduction

In this paper we present an overview of a system we have developed for generating short-term production schedules for a large steel manufacturer. The scheduling of steel production involves solving two related problems for the upstream and downstream processes of manufacturing. The downstream problem involves the selection and sequencing of groups of contiguous jobs on a number of machines subject to shift-level capacity constraints on the number of orders of each product type, tight inventory constraints and sequence-dependent setup times. The upstream processes are scheduled on unary capacity resources subject to alternate recipes and resources for each job and precedence constraints with tight minimum and maximum time lags. We have investigated a decomposition-based approach that uses mixed-integer programming and constraint programming to schedule the downstream and upstream processes respectively. One drawback to this approach is that bottlenecks on the upstream processes are not taken into account during the downstream process scheduling. We discuss some techniques we have explored to take into account such bottlenecks in a decomposition-based approach to steel production scheduling.

In the sections that follow, we present an overview of the processes in steel manufacturing and discuss the formulation of the scheduling model. We then present the two-stage decomposition based approach and discuss integration strategies to improve its performance in the presence of upstream bottlenecks. Finally, we present some experimental results.

## 2. Steelmaking processes

The scope of the scheduling problem addressed by the system covers the production of solid steel slabs from molten iron. The main processes involved in the manufacturing of steel slabs are illustrated in Figure 1. The four main process areas are:

1. The *Blast Furnace* (far left in Figure 1) 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. This flow is given as input to the scheduling problem formulation.
2. The *Basic Oxygen Furnace* is the first process that all production must pass through after leaving the blast furnace. At this process, molten iron starts to become differentiated with respect to grade and chemical composition.

- The *Refining Processes* consist of a number of steps such as reheating, ladle furnace and stirring. Not all production will pass through all of these steps, and these steps are not ordered (the steps that are used depend on the chemical composition of the final products.)
- The *Continuous Casters* (far right in Figure 1): All production passes from the refining stage to the final continuous casting process. 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 solidifies into slabs of a specific dimension.

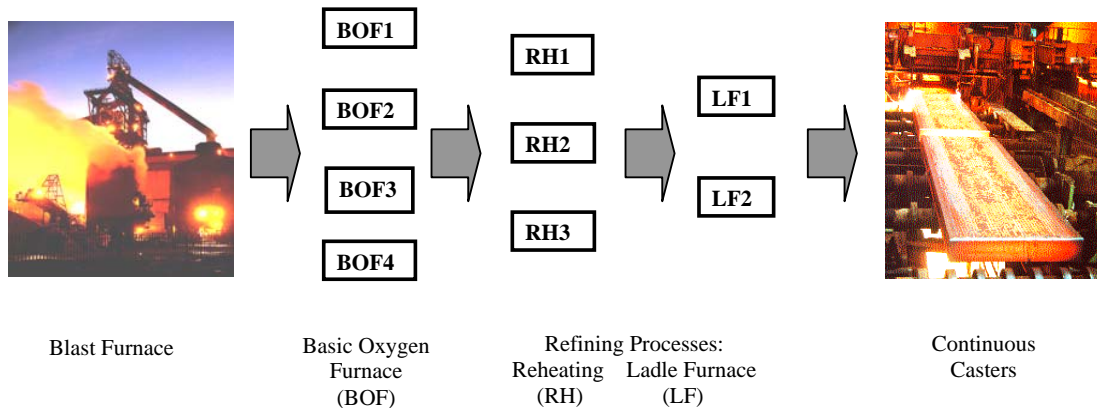


Figure 1: An illustration of the basic processes involved in the manufacturing of steel.

### 3. Scheduling model formulation

#### 3.1 Components of the model

Steel is usually produced on a make to order basis. During the manufacturing of steel products, orders for steel slabs are batched into units of “charges”. All distinct operations from the basic oxygen furnace and the refining process stages take place on a single charge of steel. At the continuous casting stage, operations take place on a sequence of (4-12) contiguous charges, which is called a “cast”. The batching of orders into charges and the sequencing of charges into casts is provided as input to the scheduling system. These batching and sequencing steps are tackled as complex, multi-criteria optimization problems, the descriptions of which are outside of the scope of this paper.

#### 3.2 Model of a single cast

Figure 2 presents a Gantt chart view illustrating how the concept of charges and casts are reflected in the formulation of the scheduling model. The Figure shows the schedule for the operations involved in the production of a single cast of steel. The interesting aspects of this formulation to note are that:

- Each set of operations, for example  $A_1$ ,  $A_2$ ,  $A_3$  and  $A_4$  in Figure 2, represent the set of operations required to produce a single charge of steel. This corresponds to a single job in the scheduling model. All activities are non-preemptible.
- In the final casting process a cast is produced consisting of a sequence of contiguous charges. This sequence is given as input to the problem. The processing of consecutive charges in a single cast on the casting processes must be without interruption. In Figure 2, the operations  $A_4$ ,  $B_3$  and  $C_2$  are scheduled as a cast on the casting process. Hence the start time of operation  $B_3$  must occur at the end time of operation  $A_4$ , and the start time of operation  $C_2$  must occur at the end time of operation  $B_3$ .

- There are tight minimal and maximal time lag constraints between consecutive operations in the same job<sup>1</sup>. For instance the maximum time lag between the end of  $A_1$  and start of  $A_2$  is 20 minutes.
- At each process there are a number of machines that can be used to process an operation. The scheduler determines which machine an operation is assigned to<sup>2</sup>. Each machine has different operating characteristics and a different physical location. As a result, the processing time of an operation at a process will depend on which machine it is assigned to.
- For each charge we are given a preferred recipe specifying a sequence of process steps that the charge must pass through. We are also given between 0 and 3 alternate recipes that can be used. In practice, most (90-95%) of charges will be assigned to their preferred recipes.

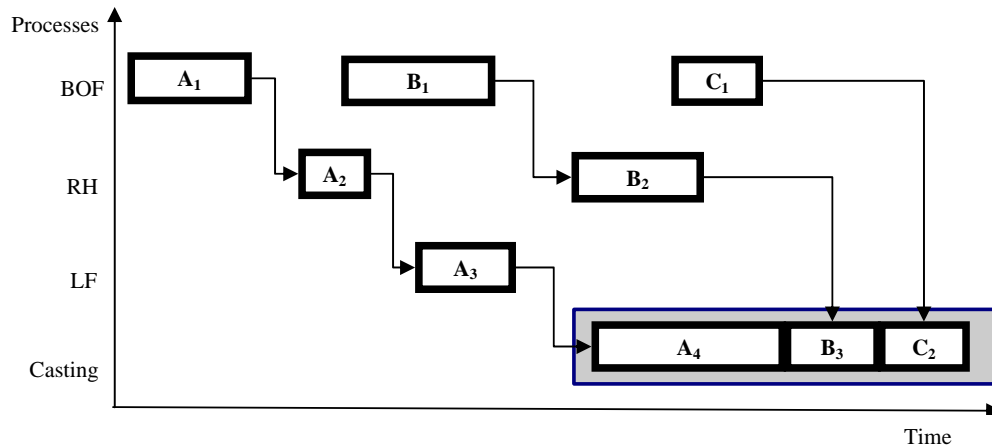


Figure 2: A Gantt chart illustration of the operations (activities) involved in manufacturing a single cast in a steel plant, from the basic oxygen furnace (BOF) to the refining processes (reheating (RH), and ladle furnace (LF)) to the casting process.

### 3.3 Hot metal inventory constraints

As we mentioned earlier, there is a continuous flow of molten iron from the blast furnace. This is specified as a problem input in terms of the number of tons per hour of molten iron flow. We consider some quantity of molten iron to be consumed in the first operation of each job at the basic oxygen furnace process. Between the blast furnace and the basic oxygen furnace there is finite capacity buffer, where the molten iron is stored until some operation is scheduled that consumes it. We have minimum and maximum inventory level constraints on the quantity of molten iron that can be allowed to accumulate in this buffer.

### 3.4 Model of cast scheduling

In section 3.2 we presented the basic scheduling model for the production of a single cast. In the full problem we are required to schedule many casts (60-100) on a number (3-9) of distinct casting machines over a 1-2 day horizon. The system is given as input a larger set of casts than can be scheduled within the horizon. The scheduler determines which casts to schedule in the short term subject to:

- Shift level capacity constraints: each charge has attributes such as product type and grade. Constraints state the minimum and maximum number of charges that can be produced per shift by each attribute. (A shift is a period of 8 hours, and there are 3 shifts per day.)

<sup>1</sup> These arise as a result of the movement of the molten steel between processes. If the steel cools down, it is necessary to reheat it, which is expensive in terms of energy consumption.

<sup>2</sup> There is an exception for the casting process, where every charge in a cast executes on the same casting machine that is given as input to the scheduler.

2. Preferred start dates: we are required to maximize the number of charges that are processed on their preferred dates.
3. Sequence dependent setup times between consecutive casts processed on the same casting machine.
4. Minimum and maximum hot metal inventory level constraints.

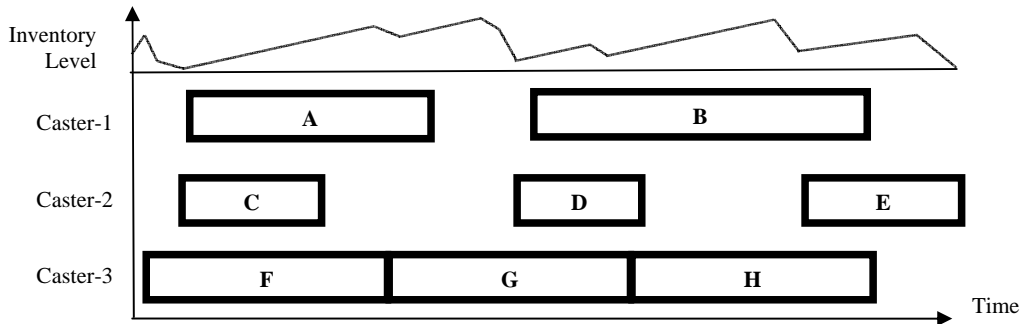


Figure 3: Illustration of a Gantt chart specifying a cast schedule.

The cast schedule specifies which casts are to be processed at what time on the available casting machines over the horizon of the schedule subject to constraints described in this section. Figure 3 illustrates a simple cast schedule for 3 casting machines. Note that on caster-3 it is possible to schedule three casts *F*, *G* and *H* consecutively with no setup time between them. Although there is no explicit objective to minimize setup time used in the schedule, in practice the maximum hot metal inventory constraint and constraints on the minimum number of charges to schedule per shift require us to use as little setup time as possible.

## 4. Solution Approach

### 4.1 Basic approach overview

The solution approach we have developed exploits the fact that we have fairly tight maximum time lag constraints between all consecutive pairs of operations in a single job for a charge (this may be as little as 20-40 minutes.) As a result, the schedule for a single cast over all processes will necessarily be localized in time. Our solution approach decomposes the full problem into two sub-problems that are solved sequentially:

1. **Downstream cast scheduling:** Cast scheduling determines which casts we are going to schedule and when; satisfying hot metal inventory constraints, shift level capacity constraints, sequence-dependent setup times between casts and preferred start times of casts. We do not consider the scheduling of any upstream processes of casting at this stage. We model this problem using a time-indexed integer programming formulation with a time granularity of 15-30 minutes and solve it using ILOG CPLEX<sup>3</sup>.
2. **Upstream detailed scheduling:** From the solution of the cast-scheduling problem we create a constraint-programming model for scheduling the processes upstream of casting, taking into account detailed scheduling constraints and preferences (such as minimum and maximum time lag between operations, resource exclusion constraints, state resource constraints, preferences on recipe and resource assignments). This model is formulated at a fine level of time granularity (30 seconds). Since the cast schedule has already been determined, we do not need to take into account any of the cast scheduling constraints in this model<sup>4</sup>.

<sup>3</sup> For the purposes of cast scheduling, we assume all charges will be scheduled on their preferred route. The upstream detailed scheduling stage may reassign routes.

<sup>4</sup> The detailed scheduler is developed using a C++ constraint-programming library for manufacturing scheduling (currently called the “Watson Scheduling Library”) developed at IBM Research.

The assumption we are making for this decomposition to work is that since the schedule for a single cast over all processes is localized in time (due to the maximum time lag constraints within a job), we can create the cast schedule for all casts independently of the processes upstream of the caster, and then use constraint programming to find a feasible upstream process schedule.

## 4.2 Basic cast scheduling integer programming model

We use a time-indexed formulation to generate a cast schedule, modeling the shift level capacity constraints and inventory constraints as side constraints. We divide the scheduling horizon into a set of equal, contiguous time periods (of usually between 15 and 30 minutes). Each time period is labeled by an index  $t$  taking a value from 1 to  $T$ . Let  $x_{it}$  be a decision variable whose value is 1 if cast  $i$  is scheduled to start processing in time period  $t$ , and whose value is 0 otherwise. Let  $p_i$  denote the number of time periods required to process cast  $i$  and let  $r_{ij}$  indicate the number of setup time periods between cast  $i$  and cast  $j$  if cast  $i$  directly follows cast  $j$  in the schedule. For each cast  $i$  we pick at most one starting time period:

$$\sum_{t=1}^{T-p_i+1} x_{it} \leq 1 \quad \forall i \quad (x_{it} = 0 \quad \forall t > T - p_i + 1) \quad (1)$$

For each caster  $C$ , given a set  $S(C)$  of casts assigned to the caster, we can process at most one cast during each time period:

$$\sum_{i \in S(C)} \sum_{s=t-p_i+1}^t x_{is} \leq 1 \quad \forall t \in \{1..T\} \quad (2)$$

The sequence-dependent setup time between each pair of casts  $i$  and  $j$  assigned to the same caster is modeled as follows:

$$\sum_{s=t+p_i}^{t+p_i+r_{ij}+1} x_{js} \leq 1 - x_{it} \quad \forall i \forall j, i \neq j \quad \forall t \in \{1..T-p_i-p_j+1\} \quad (3)$$

We introduce indicator variables  $y_{ikts}$  associated with each cast  $i$  and time period  $t$ , having the value 1 if the  $k^{\text{th}}$  charge in cast  $i$  starts processing in time period  $s$  ( $s \geq t$ )<sup>5</sup>. The hot metal inventory constraint (stating that the inventory level is maintained within some limits) is modeled as follows<sup>6</sup>:

$$\begin{aligned} HM_{\min,t} &\leq I_t \leq HM_{\max,t} \quad \forall t \in \{1..T\} \\ I_t &= I_{t-1} + P - \sum_{s \in R(t)} y_{ikts} \times W_{ik} \quad \forall t \in \{1..T\} \end{aligned} \quad (4)$$

$I_0$  is the initial inventory level and  $P$  is the hourly production rate of inventory.  $R(t)$  denotes the set of casts whose processing time periods intersects the time period  $t$ .  $W_{ik}$  is the amount of inventory consumed by the  $k^{\text{th}}$  charge of cast  $i$ . Finally, given a set of product types  $PT$ , where each product type  $\alpha \in PT$  represents the set of jobs (charges) of product type  $\alpha$ , and a set of shifts  $S$ , where each shift  $\sigma \in S$  represents the set of time periods in shift  $\sigma$ , we model the capacity constraints on the minimum ( $N_{\alpha\sigma, \min}$ ) and maximum ( $N_{\alpha\sigma, \max}$ ) number of jobs of product type  $\alpha$  in shift  $\sigma$ :

$$N_{\alpha\sigma, \min} \leq \sum_{i,k \in \alpha} \sum_{t \in \sigma} y_{ikts} \leq N_{\alpha\sigma, \max} \quad \forall \alpha \in PT, \forall \sigma \in S \quad (5)$$

The objective is usually to minimize the hot metal inventory level (subject to constraint 4).

## 4.3 Constraint programming model

Given a cast schedule, the goal of the detailed scheduler is to schedule all upstream processes taking into account detailed constraints and preferences<sup>7</sup>. A thorough description of the detailed

<sup>5</sup> Note if  $x_{it}$  is 0 then  $y_{ikts}$  is 0. This indicator variable is not strictly required (it can be directly deduced from each  $x_{it}$ ) but is introduced here to help clarify the discussion.

<sup>6</sup> The constraint presented here models hot metal inventory being consumed at the casting process, in order to simplify the presentation. In the full model, we take into account the fact that the inventory is consumed further upstream by taking into account the lead-time between casting and the upstream processes.

scheduler is presented in [4]. To summarize, the scheduler is required to determine the recipes for each charge to follow in the schedule, the resources used by each job at each process stage, the sequencing of operations on each resource at each process stage subject to unary resource capacity constraints and temporal constraints, taking into account preferences on recipe and resource assignments. The scheduling solver is based on fairly well known techniques in constraint programming, such as those presented in [1,5]. The branching in the solver is based on the precedence constraint-posting framework described in [3] for sequencing operations, and simple texture-measurement based heuristics for recipe and resource assignment selection [2]. We use timetable and disjunctive resource constraint propagation [1]. Temporal constraint propagation is performed using a variation of the incremental longest-paths algorithm developed in [6].

#### 4.4 Integration issues

One drawback of our decomposition-based approach arises from not taking into account upstream processes in the formulation of the downstream cast-scheduling problem; we encountered unforeseen bottlenecks on some of these upstream processes during detailed scheduling. Sometimes this results in the solution to the cast scheduling solution being infeasible on the upstream processes.

One solution is to extend the cast-scheduling model to perform some scheduling of the upstream processes. However, the time-indexed formulation of the cast-scheduling problem is at a relatively coarse level of time granularity (15-30 minutes), relative to the time granularity of the detailed scheduling constraints (at the 30 second level.) Using a finer time granularity in the cast-scheduling model in order to accommodate such constraints significantly increases the size of the model and the time taken to find a solution.

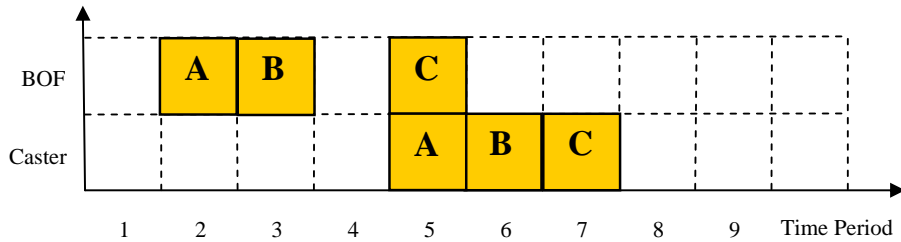


Figure 4: An illustration of the estimated capacity utilization on the upstream process BOF for a cast starting on the Caster process in time period 5.

We developed an alternative approach to avoiding upstream bottlenecks during cast scheduling by adding capacity constraints on the upstream processes to the cast-scheduling integer programming model. In order to formulate such capacity constraints, we need to be able to estimate for each cast how much capacity of the upstream processes they will utilize, and when. This is illustrated in Figure 4, where we represent part of the time-indexed cast-scheduling formulation involving a single cast of 3 charges, *A*, *B* and *C*, starting in time period 5 on the Caster and using 3 time periods (5-7) of Caster capacity. In this example if charge *A* in the cast starts in period 5 on the Caster, we estimate that it will use 1 time period of upstream BOF capacity in period 2. Note that upstream scheduling may later determine that the actual BOF capacity used by these charges is somewhere else in the schedule. However, since we have tight maximum time lag constraints between consecutive operations in a job for each charge, our working assumption is that we can estimate upstream capacity utilization that is accurate with respect to the final upstream schedule.

We generate a capacity utilization profile for each of the upstream bottleneck processes used by each of the charges in each cast<sup>8</sup>. From the capacity utilization profiles we can formulate a function  $S(i, s, t, Pr)$  which specifies the set of charges in cast *i* that (are estimated to) utilize some upstream process capacity *Pr* in time period *s*, given that the cast is scheduled to start processing in

<sup>7</sup> In practice we allow the detailed scheduler some flexibility to move casts on the casters subject to the cast sequence determined by the high level planning.

<sup>8</sup> Note that the processes upstream of casting do not have sequence dependent setup times.

time period  $t$  ( $s \leq t$ )<sup>9</sup>. We add capacity constraints to the cast-scheduling integer programming formulation for each time period  $t$  and upstream process  $Pr$  as follows:

$$\begin{aligned}
 y_{ikts} &= x_{it} && \forall i \forall s, t \in \{1..T\} \forall k \in S(i, s, t, PR) \forall Pr \\
 \sum_i \sum_{t=1}^{T-p_i+1} \sum_{k \in S(i, s, t, Pr)} y_{ikts} &\leq Cap(Pr, s) && \forall s \in \{1..T\} \forall Pr
 \end{aligned} \tag{6}$$

The term  $Cap(Pr, s)$  denotes the available capacity of the upstream process  $Pr$  in time period  $s$ .

We investigated two ways of estimating upstream capacity utilization. The first estimate (which we refer to as *min-lead-time*) is based on calculating the minimum lead-time for each charge in a cast between the upstream processes and casting process. For instance in Figure 4 this lead-time is 3 periods for charge *A* and is 1 period for charge *C*. We then construct a capacity profile based on the assumption that the lead-time is minimized between processes. An example of a capacity profile for real cast data using the *min-lead-time* estimate is illustrated in Figure 5.

The second estimate (which we refer to as *detailed-schedule*) calculates a measure of expected capacity utilization by generating a detailed schedule for each cast over all processes upstream of casting, independent of all other casts in the problem. This single cast schedule is found using the constraint programming based solver for upstream scheduling. An example of a capacity profile using the *min-lead-time* estimate for the same cast data as was used in Figure 5 is illustrated in Figure 6. Note that the *detailed-schedule* estimate here finds a capacity estimation that uses a slightly greater time extent than *min-lead-time*, but never uses more than 1 unit of capacity.

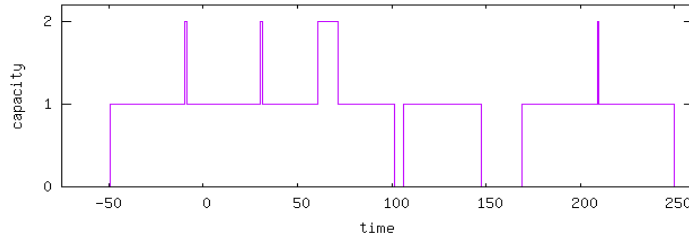


Figure 5: Min-lead-time upstream capacity estimation for a single cast.

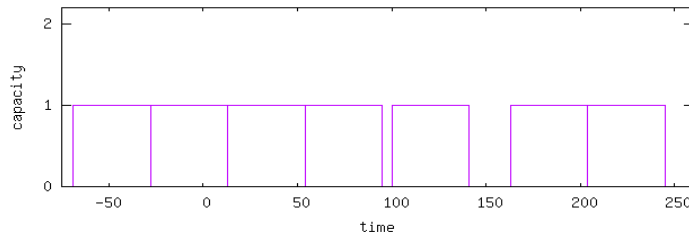


Figure 6: Detailed-schedule upstream capacity estimation for a single cast.

## 5. Experimental results

We present experimental results in Table 1 for 3 instances of real production data, comparing the *min-lead-time* and the *detailed-schedule* capacity utilization estimations. We compare the number of jobs scheduled by the cast scheduler over a 2-day horizon and the CPU time required by the cast scheduler to generate each plan. For these data sets, the upstream detailed scheduler was able to find a feasible schedule for all the casts in the cast schedule with a CPU time of usually less than 1 minute. Using the *detailed-schedule* capacity estimation, we were always able to find schedules with more jobs (charges) in the horizon than were found using the *min-lead-time* capacity estima-

<sup>9</sup> In practice it is usually sufficient to generate a single capacity utilization profile for a period of time, such as a shift, where the capacity constraints in the schedule do not change.

tion, using less CPU time. In practice, when using the *min-lead-time* capacity estimation we were usually not able to match the performance of the human experts in generating production schedules, with respect to number of jobs scheduled. With the *detailed-schedule* capacity estimation, we were able to significantly improve upon the performance of the human experts.

Instance	Number of Jobs (Charges) Scheduled		CPU Time (sec) for Cast Scheduling	
	min-lead-time	detailed-schedule	min-lead-time	detailed-schedule
1	122	138 (11.59%)	1406	160
2	162	163 (0.61%)	596	215
3	113	134 (15.67%)	158	146

Table 1: Experimental results for cast scheduling.

## 6. Conclusions

The scheduling of steel production involves solving two related problems for the upstream and downstream processes of manufacturing. The downstream, cast-scheduling problem requires the selection and sequencing of groups of contiguous jobs (casts) on a number of machines subject to shift-level capacity constraints, inter-process inventory constraints and sequence-dependent setup times. The upstream processes are scheduled on unary capacity resources subject to alternate recipes and resources for each job and precedence constraints with tight minimum and maximum wait times. We have presented a decomposition-based approach that uses mixed-integer programming to generate a production plan for downstream cast scheduling at a coarse level of time granularity, and constraint programming to schedule upstream processes subject to detailed scheduling constraints at a fine level of time granularity. In order to improve the performance of this approach, we found it necessary to take into account bottlenecks, which could appear on the upstream processes into the downstream planning model. We developed an approach for adding constraints to this downstream model by first generating a detailed upstream schedule for each cast in isolation using constraint programming, and using such schedules to estimate the capacity utilization for each cast on the upstream processes. We have presented a small set of experimental results that indicate this approach shows some promise.

## Acknowledgements

The authors would like to thank Fatma Kilinc Karzan for comments on earlier drafts of this paper.

## References

- [1] P. Baptiste, C. LePape and W. Nuijten (2001), *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*, International Series in Operations Research and Management Science, Vol. 39, Kluwer.
- [2] J.C. Beck, A.J. Davenport, E.M. Sitariski and M.S. Fox (1997), Texture-Based Heuristics for Scheduling Revisited, *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, 241-248, AAAI Press / MIT Press.
- [3] C.C. Cheng, and S.F. Smith (1996). Applying constraint satisfaction techniques to job shop scheduling, *Annals of Operations Research, Special Volume on Scheduling: Theory and Practice*, 1.
- [4] A. Davenport, J. Kalagananam, C. Reddy, S. Siegel and J. Hou (2007), An application of constraint programming to generating detailed operations schedules for steel manufacturing, *Proc. 13<sup>th</sup> International Conference on Principles and Practice of Constraint Programming*.
- [5] U. Dorndorf, E. Pesch, and P.-H. Toan, A Time-Oriented Branch-and-Bound Algorithm for Resource-Constrained Project Scheduling with Generalised Precedence Constraints”, *Management Science* **46**(10), 1365 – 1384.
- [6] I. Katriel and P. Van Hentenryck (2005), Maintaining Longest Paths in Cyclic Graphs, *Proc. 11<sup>th</sup> International Conference on Principles and Practice of Constraint Programming*.