An Application of Constraint Programming to Generating Detailed Operations Schedules for Steel Manufacturing

Andrew Davenport[1], Jayant Kalagnanam[1], Chandra Reddy[1], Stuart Siegel[1], and John Hou[2]

[1] IBM T.J.Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598, USA
[2] IBM Taiwan Business Consulting Services, Shong-Ren Rd, Taipei, Taiwan, Republic of China
davenport@us.ibm.com, jayant@us.ibm.com, creddy@us.ibm.com, ssiegel@us.ibm.com, kwhou@tw.ibm.com

Abstract. We present an overview of a system developed by IBM for generating short-term operations schedules for a large steel manufacturer. The problem addressed by the system was challenging due to the combination of detailed resource allocation and scheduling constraints and preferences, sequence dependent setup times, tight minimum and maximum inventory level constraints between processes, and constraints on the minimum and maximum levels of production by shift for each product group. We have developed a domain-specific decomposition based approach that uses mixed-integer programming to generate a high-level plan for production, and constraint programming to generate a schedule at fine level of time granularity taking into account detailed scheduling constraints and preferences. In this paper we give an overview of the problem domain and solution approach, and present a detailed description of the constraint programming part of the system. We also discuss the impact the system is having with the customer on their manufacturing operations.

1 Introduction

In this paper we present an overview of a system developed by IBM for generating short-term detailed operations schedules for a large steel manufacturer. The problem addressed by the system involves generating schedules subject to detailed constraints and preferences at a fine level of time granularity (30 seconds or less). Scheduling problems with such low-level constraints and preferences are typically handled well by constraint programming techniques [1,3]. However the problem also contains constraints and objectives that are stated at a coarser level of time granularity, and might be considered to be more at the "planning level". Examples of such constraints and objectives are deciding which orders to produce over the next few days from an order pool, subject to capacity constraints on the number of orders of each product type that can be produced within each shift, and maximizing the number of orders scheduled on their preferred start date. This aspect of the problem is often handled well using mixed

C. Bessiere (Ed.): CP 2007, LNCS 4741, pp. 64 – 76, 2007.

integer programming techniques. In order to combine the advantages of both constraint programming and integer programming, we developed a hybrid solution approach that decomposes the full problem into a high level integer programming based planning problem to determine what orders to produce and roughly when on the available process stages, and a low-level constraint programming based scheduling problem to determine a detailed second-by-second schedule on specific machines at each process. However the complex global nature of some of the detailed scheduling constraints, combined with the requirement to achieve a high level of resource utilization, resulted in this decomposition-based approach producing poor solutions that were not acceptable to the customer. In order to overcome this drawback, we developed a simple integration mechanism between the planning and detailed scheduling stages that was found to significantly improve solution quality. A detailed description of the mixed integer programming formulations used in this decomposition-based approach is presented in [6]. In this paper we present an overview of the application domain and the integrated problem solving approach, and discuss in detail the constraint programming aspects of solving this problem.

2 The Problem

The scope of the problem addressed by the system is to produce a multi-day operations schedule for the manufacture of steel products from raw material inputs. The main processes involved in the manufacturing of steel are illustrated in Figure 1. The four main process areas addressed by the system 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 specified on an hourly basis.
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.
3. 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.)
4. 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.

Steel is usually produced on a make to order basis. Customer orders are batched into units of production called “charges”. All distinct operations (activities in the scheduling model) 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 (2-12) 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 form the basis of a complex, multi-criteria sequencing problem, the descriptions of which are outside of the scope of this paper, but which were part of the overall system developed by IBM for the customer.

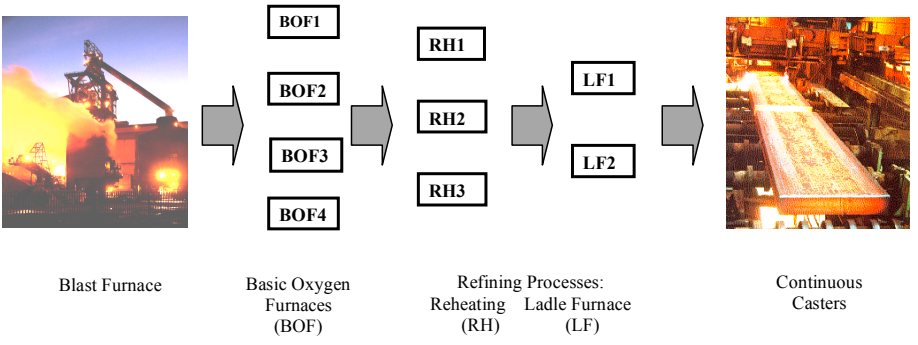

Fig. 1. An illustration of the basic process flow for the manufacture of steel

We divide the full problem into two problem stages: the downstream cast scheduling problem, which corresponds to the high-level planning problem, and the upstream processes detailed scheduling problem. We describe each of these problems stages in the sections which follow.

2.1 Detailed Scheduling Problem Model

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 activities involved in the production of a single cast of steel. The interesting aspects of this formulation to note are that:

1. Each set of activities, for example A_1, A_2, A_3 and A_4 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.
2. 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 activities A_4, B_3 and C_2 are scheduled as a cast on the casting process. Hence the start time of activity B_3 must occur at the end time of activity A_4, and the start time of activity C_2 must occur at the end time of activity B_3.
3. There are tight minimum and maximum time lag constraints between consecutive activities in the same job[1], for instance the maximum time lag between the end of activity A_1 and start of activity A_2 might be 20 minutes.

[1] Maximum time lags 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. Minimum time lags arise from the transfer time of materials between processes.

4. At each process stage there are a number of resources (machines) that can be used to process an activity (between 2 and 5). The scheduling system needs to determine which resource at each process stage each activity is assigned to[2]. Each resource has different operating characteristics and a different physical location. As a result, the processing time of an activity at a process stage, as well as the transfer time between processes, will depend on the specific resources at each process the activity is assigned to.
5. For each charge we are given a preferred recipe specifying the sequence of process steps that the charge must pass through. We are also given between 0 and 3 alternate recipes that can be used, should there not be sufficient capacity on the preferred recipe process stages. In practice, most (85-95%) of charges will be assigned to their preferred recipes.

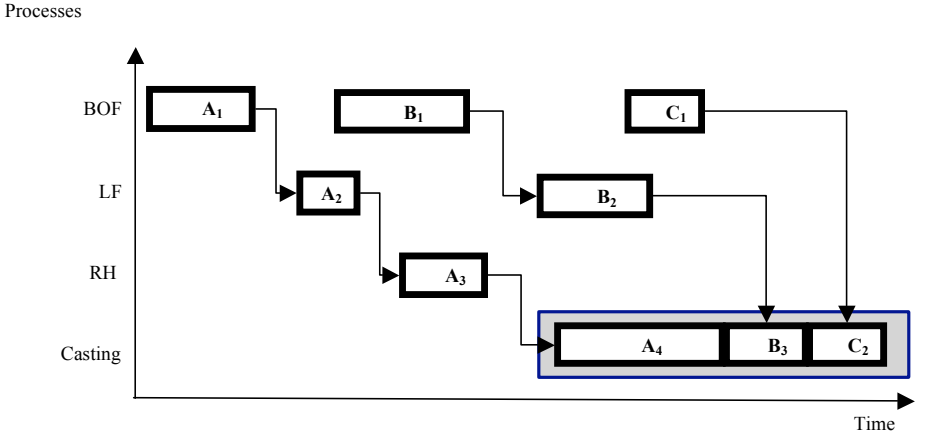

Fig. 2. A Gantt chart illustration of the activities involved in manufacturing a single cast in a steel plant, from the basic oxygen furnace (BOF) to the refining processes (reheating (RH), and ladle furnace (LF)) to the casting process

All the scheduling resources in the problem model have unary capacity (at most one activity at any time point can be executing on a specific resource.) However some resources are state resources, whose state is represented by the number of activities it has processed since the last "setup" activity on the resource. Once the resource has processed a maximum number of activities, it is required to perform another setup activity. Charges on a state resource have a range specifying the minimum and maximum values with respect to the resource state, within which they can be processed on the resource.

2.2 Cast Scheduling Problem Model

We are required to schedule between 60 and 100 casts each composed of 2-12 charges on one of a number (3-9) of distinct casting machines and upstream processes over a

[2] 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.

1-2 day horizon. The system selects which casts to schedule within the horizon from a pool of around 200 available casts given as input.

As mentioned earlier, there is a continuous flow of molten iron from the blast furnace. The amount of this flow over time is specified as a problem input in terms of number of tons per hour. We consider some quantity of molten iron to be consumed by the first activity of each job when it starts processing 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 activity is scheduled to consume it. There are tight constraints on the minimum and maximum quantity of molten iron that can be allowed to accumulate in this buffer.

The cast schedule specifies which casts are to be processed at what time on the available casting machines, subject to the following constraints:

1. Shift level target and capacity constraints: each charge has attributes such as product type and grade. Constraints state the target, 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.)
2. Each charge is associated with a preferred start date. We are required to maximize the number of charges that are processed on their preferred start 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 illustrates a simple cast schedule for three 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.

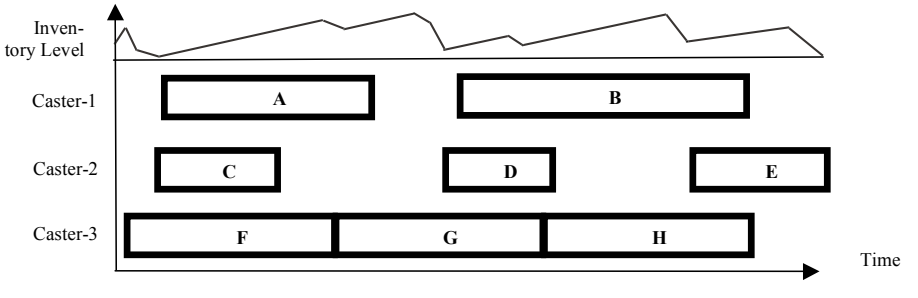

Fig. 3. An illustration of a Gantt chart that specifies a cast schedule

3 Why CP?

The system developed by IBM was designed to be used as a decision support tool for the scheduling department at a steel manufacturing plant. Prior to our involvement, all scheduling was performed manually by a group of scheduling experts. The nature of steel manufacturing is such that production planning, design and operations scheduling are generally more complex than is found in other industries. In particular, optimization problems in the steel industry often involve many constraints expressed at a very low level of detail, yet the tightness of these constraints can have a major impact on the solution at a global level. Furthermore, most steel industry problems that we have encountered contain multiple, often competing objectives. As a result, off-the-shelf supply chain tools, even with some customization, are usually not able to cope with the complexity of optimization problems found in the steel industry.

The complexity of the problem model we encountered for this problem is such that constraint programming seemed a natural choice for the detailed scheduling part of the system. Some of the detailed constraints in the problem, such as those involving state resources, would be quite difficult to model and maintain using an integer programming formulation. Even so, this problem was still quite challenging to solve using constraint programming, since some aspects of the problem have received little attention in the research literature. Examples of such aspects include scheduling with non-substitutable resource alternatives and alternative recipes [7,9] and taking into account detailed preferences on resource and recipe assignments for each activity or set of activities. (One example of the source of such preferences was that if something goes wrong during execution, the schedule should be designed in such a way that it was easy for the human experts to take out "chunks" of the schedule to reschedule elsewhere as quickly as possible.) As such, we were required to experiment and develop solution approaches for dealing with these aspects of the problem during the project. One advantage of using constraint programming, compared with integer programming approaches, is the relative ease with which the constraint programming model and solution approach could quickly and flexibly accommodate change requests over the lifespan of the project.

Although constraint programming seemed like a good choice of technology for the detailed scheduling aspect of the problem, the cast scheduling problem contains constraints and objectives that are more amenable to an integer programming approach. As such, the solution approach we developed decomposes the full problem into two sub-problems that are solved in sequence:

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[3].

[3] 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.

2. ***Upstream process 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 lags between activities, 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[4].

This solution approach exploits the fact that we have fairly tight maximum time lag constraints between all consecutive pairs of activities 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 both on the casting process and the upstream processes.

4 How CP?

We solve the cast scheduling problem using mixed-integer programming, formulating the problem using a time-indexed formulation and modeling the shift level capacity constraints, the sequence dependent setup times and the hot metal inventory constraints as side constraints. The scheduling horizon in the time-indexed model is divided into a set of contiguous time periods of equal size (between 15 and 30 minutes.) We present the full mixed integer programming model in [6]. In the sections that follow, we discuss in detail the use of constraint programming in the overall system and give an overview of the integration between the integer programming model and the constraint programming model.

4.1 Constraint-Programming Detailed Scheduling Solver

Given a cast schedule that specifies which casts are to be scheduled in the horizon and an approximate starting time for each selected cast, the goal of the detailed scheduler is to schedule all processes upstream of casting, subject to the following:

1. Select a recipe from a number of available alternate recipes for each charge in each cast to follow in the schedule;
2. Select a resource (machine) from a number of available non-substitutable resources for each job at each process stage;
3. Assign start times to activities on each resource at each process stage, subject to unary resource capacity constraints and precedence constraints with minimal and maximal time lags;
4. Take into account preferences on alternate recipe and resource assignments.

[4] The detailed scheduler is based on a C++ constraint-programming library for manufacturing scheduling developed by IBM (currently known as the "Watson Scheduling Library").

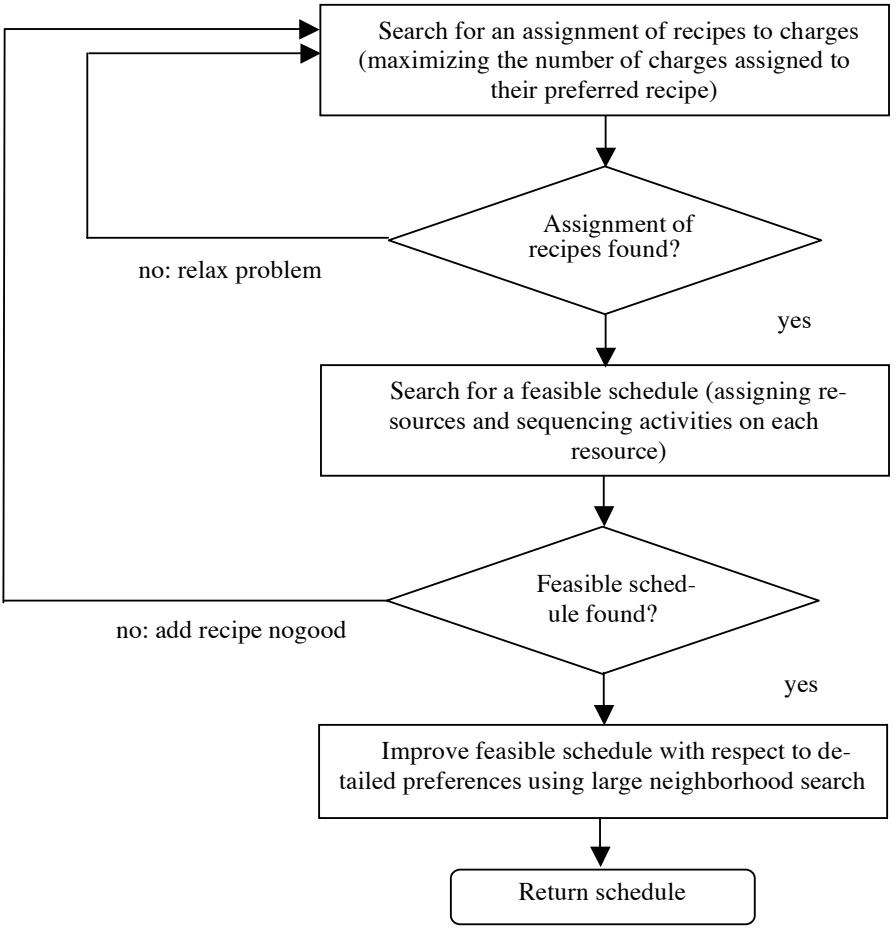

Fig. 4. Problem solving flow for detailed scheduling

We use constraint programming to perform detailed scheduling [1]. An outline of the constraint programming approach we use to perform detailed scheduling is presented in Figure 4.

The problem solving process takes place in several stages. Firstly, we search for a complete feasible assignment of recipes to charges. We perform depth-first search with chronological backtracking. We use a chronological variable ordering heuristic, following the initial start time for each cast specified by the cast scheduling solution. Alternate recipes for each charge are tried in order of preference. Usually we can find an assignment of recipes that uses the most preferred recipe for each charge. In rare cases that a feasible assignment of recipes cannot be found, we relax the problem by selecting a cast to remove from the schedule.

The next stage is to find a feasible schedule that for each recipe, assigns a resource to each activity at each process stage in the recipe, and sequences all activities on

each resource. The search approach for finding a feasible schedule is based on the precedence constraint-posting framework described in [3], using chronological backtracking and some simple texture-measurement based heuristics for resource assignment selection [2]. We use the timetable and disjunctive resource constraint propagators [1,5]. The difficulty in solving this problem arises more from making the right choices of resources to use for each activity to satisfy the precedence constraints with tight maximal time lags, rather than in sequencing the activities on each resource. As such, we did not find complex constraint propagation approaches such as edge-finding [1] to be useful for solving this problem. Temporal constraint propagation is performed using a variation of the incremental longest-paths algorithms developed in [4]. If we cannot find a feasible schedule after some backtrack limit is reached, we identify a recipe to add as a nogood recipe and return to the recipe assignment stage.

Once we have a feasible schedule, we attempt to improve the quality of the solution with respect to preferences on resource and recipe assignments. For this, we have many detailed preferences specified as rules by the user. For example, one such rule might specify that all charges in the same cast should try to use the same resource in the reheating process. We use constraint programming based large neighbourhood search to perform this improvement phase [11].

In practice, the detailed scheduler is quite fast: an initial feasible schedule can usually be found in less than 5 seconds on a 1.6 GHz Pentium 4 laptop. Improving the schedule using large neighborhood can take 2-3 minutes. The cast scheduler is the most time consuming part of the system: finding a cast schedule within 1% of optimality with CPLEX usually takes 5-10 minutes.

4.2 Integration Issues

One drawback of the decomposition-based approach we have presented arises from not taking into account upstream processes in the formulation of the downstream cast-scheduling problem. We sometimes encountered unforeseen bottlenecks on some of these upstream processes during detailed scheduling based on the cast schedule solution. Sometimes this results in the solution to the cast scheduling problem found by the integer programming solver being infeasible on the processes upstream of casting.

One solution to this problem is to extend the cast-scheduling model to perform some scheduling of the upstream bottleneck 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 and complexity of the model and the time taken to find a solution.

We developed an alternative approach to avoiding upstream bottlenecks by adding capacity constraints on the upstream bottleneck processes as side constraints 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 are expected to utilize, and when. This is illustrated in Figure 5, 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 might estimate that it will use 1 time period of upstream BOF capacity in period 2. Note that later detailed scheduling of the upstream processes may determine that the actual BOF capacity used by these charges is somewhere else in the schedule. However, since we have tight maximum wait time constraints between consecutive activities in a job for each charge, our working assumption is that we can estimate upstream capacity utilization for each cast that is fairly accurate with respect to the final upstream schedule.

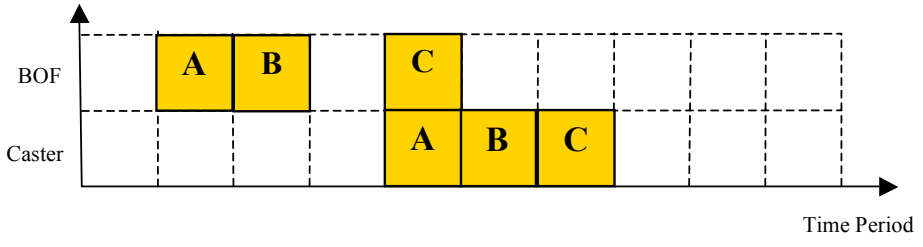

Fig. 5. An illustration of the estimated capacity utilization profile on the upstream process BOF for a cast startin g on the Caster process in time period 5

More specifically, for each cast and each time period *t* in the time-indexed integer programming model and for each upstream bottleneck process, we estimate the capacity used by the cast on the process if it starts processing in time period *t* on the casting process. (In practice it is not necessary to estimate this for every time period: it is sufficient to generate a single estimation for all time periods where the capacity constraints do not change.) We use this estimation as a basis to formulate the upstream process capacity constraints to add to the time-indexed integer programming model. For such an estimation to be useful, it should take into account the detailed scheduling constraints on the upstream processes. We do this by generating a detailed schedule for each cast on all upstream processes, independently of all other casts, using the constraint programming scheduling solver. We use the solution generated by this solver as the basis to estimate upstream capacity utilization for each cast.

Experiments on customer problem data show that by using capacity constraints generated from a constraint programming solution for a single cast can improve both the quality of the final schedule with respect to number of orders scheduled, and well as speed up execution time of the solver. We present the exact formulation of these capacity constraints, as well as experimental results comparing the performance of constraint-programming generated capacity constraints to those generated using simple heuristics, in [6].

We summarize the full high-level problem solving flow used by the system for a generating a steel-making schedule in Figure 6.

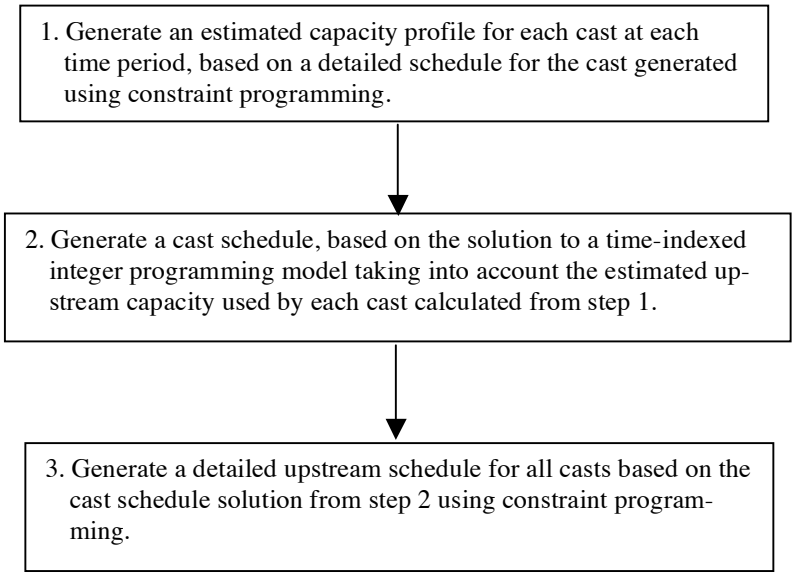

Fig. 6. High-level solution approach process flow using constraint programming and mixed integer programming

5 Added Value of CP?

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 by two researchers at IBM 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. The impetus for the user to improve their scheduling processes arose from the growing pressure from their customers to improve on-time delivery and provide shorter ordering promising time. 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 unsatisfactory.

As of writing, the system has just started to go into use by the end user, in parallel with their current system. Initial feedback has been very favourable. The users interact with the system through a graphical user interface, allowing them to influence aspects of the solution such as stating that some casts must be included in the schedule within some specified time range. The users are very impressed that the IBM system is able to generate schedules that achieve higher resource utilization, by scheduling up to 10% more charges, than that of hand-generated schedules prepared

by the expert human schedulers. Furthermore, the system can generate full 2-day schedules within 5-10 minutes on a 3GHz Opteron Linux machine, as opposed to many hours needed for the human experts to generate a schedule. This is important, since in practice the users may 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. Since the shop floor is very dynamic, some real-time adjustment of the schedules generated by the system is performed by the users during execution.

6 Related Work

Due to the nature of the manufacturing processes, the complexity of production planning and operations scheduling is usually higher in the steel industry than in many other industries. As a result, many commercial "off-the-shelf" tools cannot adequately address the full scope and complexity of production planning and scheduling in the steel industry. We believe that constraint programming can be an important component of decision-support solutions in this area. Some other applications of constraint programming in the steel industry include the system presented in [12] to perform bloom sequencing at what was formerly British Steel. The bloom-sequencing problem is an "upstream" optimization problem to the system presented in this paper: it is used to design the casts whose production is then scheduled by the steelmaking scheduling system. (A system to perform bloom sequencing (as well as plate and coil sequencing) was designed and implemented by IBM as part of the overall project, but is not described in this paper. This system uses a decomposition-based approach utilizing integer programming and specialized bin-packing heuristics [10].) Constraint programming is also used in the COORDIAL system developed using CHIP for real-time scheduling of the production of steel for the Sollac Group in France [13].

7 Conclusions

We have presented an overview of a system for generating detailed schedules for steel production that has developed by IBM for a large steel manufacturer. The full scheduling problem addressed by the system involves solving two related problems for the upstream and downstream processes of steel 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 scheduling problem involves determining a detailed schedule for processes upstream of the casting processes, taking into account preferences on how resources are allocated (such as alternate recipes and resources used by each job), precedence constraints with tight minimum and maximum time lags and complex state resources. We have presented an integrated, 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. Initial end-user feedback has been very favourable.

References

[1] Baptiste, P., LePape, C., Nuijten, W.: Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems. In: International Series in Operations Research and Management Science, vol. 39. Kluwer, Dordrecht (2001)

[2] Beck, J.C., Davenport, A.J., Sitarski, E.M., Fox, M.S.: Texture-Based Heuristics for Scheduling Revisited. In: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), pp. 241–248. AAAI Press / MIT Press (1997)

[3] Cheng, C.C., Smith, S.F.: Applying constraint satisfaction techniques to job shop scheduling. Annals of Operations Research, Special Volume on Scheduling: Theory and Practice 1 (1996)

[4] Katriel, I., Van Hentenryck, P.: Maintaining Longest Paths in Cyclic Graphs. In: Proc. 11th International Conference on Principles and Practice of Constraint Programming. Springer, Heidelberg (2005)

[5] Laborie, P.: Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results. Artificial Intelligence Journal 143(2), 151–188 (2003)

[6] Davenport, A., Kalagnanam, J.: Scheduling steel production using mixed-integer programming and constraint programming. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (2007)

[7] Beck, J.C., Fox, M.S.: Scheduling Alternative Activities. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) (1999)

[8] Kramer, L.A., Smith, S.F.: Maximizing Flexibility: A Retraction Heuristic for Oversubscribed Scheduling Problems. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03) (2003)

[9] Focacci, F., Laborie, P., Nuijten, W.: Solving Scheduling Problems with Setup Times and Alternative Resources. In: Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling Systems (AIPS 2000), pp. 92–111 (2000)

[10] Lee, H.S., Trumbo, M.: An Approximate 0-1 Edge-Labeling Algorithm for Constrained Bin-Packing Problem. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), pp. 1402–1411 (1997)

[11] Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Maher, M.J., Puget, J.-F. (eds.) Principles and Practice of Constraint Programming - CP98. LNCS, vol. 1520. Springer, Heidelberg (1998)

[12] Smith, A.W., Smith, B.: Constraint Programming Approaches to a Scheduling Problem in Steelmaking. School of Computing Research Report 97.43, University of Leeds (September 1997)

[13] http://www.cosytec.com/constraint_programming/cases_studies/steel_industry.htm