Lot-Sizing and Furnace Scheduling in Small Foundries

Silvio A. de Araujo
Instituto de Biociências, Letras e Ciências Exatas
Departamento de Ciências da Computação e Estatística
Universidade Estadual Paulista, São José do Rio Preto SP, 15054-000, Brazil
Email: saraujo@ibilce.unesp.br

Marcos N. Arenales
Instituto de Ciências Matemáticas e de Computação
Departamento de Matemática Aplicada e Estatística
Universidade de São Paulo,Caixa Postal 668, São Carlos SP, 13560-970, Brazil.
Email: arenales@icmc.usp.br

Alistair R. Clark *
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England, Bristol, BS16 1QY, United Kingdom.
Email: Alistair.Clark@uwe.ac.uk
Tel: +44 117 328 3134

* Corresponding author

Abstract: A lot-sizing and scheduling problem prevalent in small market-driven foundries is studied. There are two related decision levels: (1) the furnace scheduling of metal alloy production, and (2) moulding machine planning which specifies the type and size of production lots. A mixed integer programming (MIP) formulation of the problem is proposed, but is impractical to solve in reasonable computing time for non-small instances. As a result, a faster relax-and-fix (RF) approach is developed that can also be used on a rolling horizon basis where only immediate-term schedules are implemented. As well as a MIP method to solve the basic RF approach, three variants of a local search method are also developed and tested using instances based on the literature. Finally, foundry-based tests with a real order book resulted in a very substantial reduction of delivery delays and finished inventory, better use of capacity, and much faster schedule definition compared to the foundry's own practice.

Key words: lot-sizing and scheduling, meta-heuristics, mixed integer programming.

1. Introduction

Foundries are common in every region of Brazil, producing many types of products, ranging from simple items for domestic use to sophisticated parts for the automobile and machine tool industries. According to the Brazilian Foundry Association [1], the sector is growing rapidly, having produced over 12,000 tonnes a day (October/2005) and directly employing about 60,000 people.

Foundries can be classified as either captive or market-driven. The former tend to be part of large companies, such as car manufacturers, that totally absorb the foundry production, composed of large quantities of a small number of parts with relatively stable demand. On the other hand, market-driven foundries are generally small or medium sized companies that nevertheless produce a huge range of items with vastly varying demand. While captive foundries use a small number of different metal alloys, market-driven foundries need to work with a wide variety due to the diversity of their clients. This paper focuses on the problem of planning and scheduling production in small market-driven foundries.

Numerous researchers have studied lot sizing and setup scheduling problems, with reviews by [2-10]. However, there is little published research on such problems in foundries, and even less concerned with market-driven foundries. Santos-Meza *et al.* [11] studied the problem in small and medium-sized foundries, while Araujo and Arenales [12] researched a large captive foundry. Sounderpandian *et al.* [13] and Gravel *et al.* [14] also studied to large foundries, but dealt specifically with the problem of job sequencing on machines. Most papers on this topic are concerned with production scheduling in large steel plants. By their nature, such plants do not have moulding sections since they produce steel sheets, of varying sizes and types, in rolling machines with cylinders that are configured according to the sheet specifications. Tang *et al.* [15] published a review of research on production planning and scheduling in steel production, while a variety of papers have studied the practical problems found in large steel plants, including [16-23].

In practice, the problem is tackled sequentially. The lot sizing of foundry products is carried out first, taking into account their demand and due dates. The production of alloys is then scheduled as a function of the lot sizes of the end products. However, this approach can result in a poor furnace schedule given the lack of two-way linkage between the lot sizing of end products and the scheduling of alloys. In this paper, we propose an optimization model where the two problems are solved in an integrated manner. The model put forward in this paper is closely related to the General Lot Sizing and Scheduling Problem (GLSP) and its extensions [8, 24-28]. The GLSP schedules multiple products on a single machine and allows many setups in each single ‘large-bucket’ time period. This paper adapts the GLSP to include backlogs and product-group setups, with an emphasis on rolling horizon use.

The review by Karimi *et al.* [10] highlights the development of heuristics with reasonable speed and solution quality for this kind of model as an important research area. Recent research by Gupta and Magnusson [29] into capacitated lot sizing with sequence-dependent setup costs states that it is still difficult to obtain near-optimal solutions for industrial-size problems. Dillenberger *et al.* [30] formulated a lot sequencing and sizing model with representation of sequence-independent setup times on multiple machines. The resulting mixed integer programming (MIP) model is difficult to solve optimally for large realistic problems, and so the authors resorted to the *fix-and-relax* method (Beraldi *et al.* [31]), more widely known as *relax-and-fix* (Wolsey [32], Kelly and Mann [33]). The current paper uses a similar basic approach. Dillenberger *et al.* [30] illustrated the viability and value of the *relax-and-fix* method, applying it to sizeable real problems from three IBM plants, and obtaining acceptable solutions in reasonable computing time, even on the slower machines of the 1990s.

Section 2 describes the lot sequencing and sizing in the context of found while section 3 presents the results MIP model. An initial attempt to optimally solve this model using advanced optimisation software was not successful. To try to overcome this, section 4 presents rolling horizon solution strategy, but the software still failed to find good solutions within reasonable computational time. The relax-and-fix heuristic approach was then applied on a rolling horizon basis in section 5, giving good results. Relax-and-fix involves the solution of a period-by-period sequence of partially-relaxed MIPs, each one with just a reduced set of binary variables whose number is small enough to obtain good solutions. As the horizon rolls forward in time over the periods, each set of binary variables is permanently fixed at their solution values.

For comparison, three methods involving neighbourhood search on the reduced set of binary variables were developed, namely, descent heuristic, diminishing neighbourhood search and simulated annealing. Several authors have already explored a similar approach, among them Kuik *et al.* [34] who used simulated annealing and tabu search on a lot-sizing problem with sequence-independent setups, Teghem *et al.* [35] who employed linear programming within a simulated annealing for a combinatorial production planning problem, and Fleischman and Meyr [26] who applied threshold accepting search on the general lot sizing problem (GLSP) with sequence-independent setup costs (but not times), denoted GLSPST. Similarly, Meyr [27] developed an efficient algorithm for the GLSPST that uses multiple runs on a local search for lot sequencing with linear programming (LP) dual reoptimization for rapid lot-sizing.

Computational tests comparing all the methods are presented and analysed in section 6, including in-foundry comparisons with practiced schedules.

2. Problem Definition

A key process in a foundry is the transformation of ore and scrap metal into alloys with specified levels of carbon, silicon, zinc, etc, that determine properties such as brittleness and

resistance to corrosion. The alloy, still in a liquid state, is then poured into moulds, normally made of sand and resin where it cools to produce final items. These two processes of alloy production and item moulding must be jointly scheduled. After the cast items have been cooled, they are deburred and made available for delivery.

In small foundries, only one furnace is usually operating at any point in time, so that just a single alloy can be produced in each time period. This is different from large foundries where several furnaces can be in operation simultaneously, enabling the production of several alloys in the same period [12]. Furthermore, in small foundries, the preparation of sand moulds is a manual process that is carried out as soon as the next day's production schedule is specified and is not a production bottleneck. Rather, the furnace is the production bottleneck. This contrasts with automated foundries, where multiple moulding machines of varying capacity and efficiency have to be scheduled. In this case, the bottleneck could be either mould preparation or the furnaces, depending on item orders and schedules.

Figure 1 illustrates the main activities in a small market-driven foundry. Clients randomly and spontaneously submit orders specifying the item type, quantity and alloy. The Production Planning department negotiates due dates with the client, often agreeing unachievable dates that result in delivery delays and the possible loss of future orders from the client. Thus the minimisation of delays is one of the principal concerns of the foundry company.

The Production Planning department specifies which items should be produced during the next few days, advises the Moulding section which moulds to prepare in advance, and determines the specific alloys to be melted. The scheduling is guided by due dates in the following days as well as by delayed orders. Excessive changes of alloys are undesirable, and so setups are considered in the model.

The manufacturing system has the following characteristics and assumptions:

- An alloy is generally used in several products, but a product is made from just one alloy.
- The output weight of an alloy is equal to the total gross weight of the products in which the alloy is used.
- A product cannot be manufactured in a given time period unless the alloy which it is made from is also processed in that period. Processed alloys cannot be held over to the next period.
- In each time period only one alloy can be processed on the furnace.
- A setup changeover from one alloy to another consumes capacity time in a manner that is independent of the sequence in which the alloys are processed.
- All products have a demand over a planning horizon that would be met if capacity were sufficient. However, delays will occur if capacity is tight, and so backlogs must be represented in the model.
- The objective is to schedule lot sizes and to sequence setups in order to minimize a penalty-weighted sum of product backlogs, finished inventories and setup changeovers.

3. Mathematical Model

We now propose a mixed integer programme (MIP) to model the problem described above. The following notation is used:

Indices:
- $k = 1, ..., K$ alloys
- $i = 1, ..., N$ items
- $t = 1, ..., T$ periods (days, for example)
- $n = 1,...,h$ sub-periods (furnace loadings lasting 2 hours, for example)

Data:

Cap	Capacity (kg) of a single furnace loading
r_i	Gross weight (kg) of item i
d_{it}	Quantity of items i ordered for period t
$S(k)$	Set of items i that use alloy k (each item uses one and only one alloy). Thus $\{1, ..., N\} = S(1) \cup ... \cup S(K)$ and $S(k) \cap S(j) = \varnothing$ for all $k \neq j$
h_{it}^-	Penalty for delaying a unit of item i in period t
h_{it}^+	Penalty for holding a unit of item i in period t
s_k	Setup penalty for alloy k
st_k	Setup loss of capacity (kg) due to a setup for alloy k

Variables:

x_{in}	Quantity (lot-size) of item i to be produced in sub-period n
I_{it}^+	Quantity of item i held at the end of period t
I_{it}^-	Quantity of item i delayed at the end of period t
y_n^k	Binary variable, $y_n^k = 1$ indicates that the furnace is set up (configured) for producing alloy k in sub-period n, otherwise $y_n^k = 0$
z_n^k	Binary variable, $z_n^k = 1$ if there is a setup (changeover to) alloy k in sub-period n, otherwise $z_n^k = 0$. Thus $z_n^k = 0$ if $y_{n-1}^k = y_n^k$ and $z_n^k = 1$ if $y_{n-1}^k < y_n^k$.

Furthermore, consider the following definitions from the General Lot-Sizing and Scheduling Problem (GLSP) model [8, 24-28]:

h_t	Number of sub-periods in period t
$F_t = 1 + \sum_{t=1}^{t-1} ?_t$	First sub-period in period t ($F_1 = 1$)
$L_t = F_t + h_t - 1$	Last sub-period in period t
$h = \sum_{t=1}^{T} ?_t$	The total number of sub-periods over the planning horizon

We can now formulate the following MIP model:

Minimize $$\sum_{i=1}^{N}\sum_{t=1}^{T}(h_{it}^{-}I_{it}^{-} + h_{it}^{+}I_{it}^{+}) + \sum_{k=1}^{K}\sum_{n=F_1}^{L_T}(s_k z_n^k) \qquad (1)$$

subject to:

$$I_{i,t-1}^{+} - I_{i,t-1}^{-} + \sum_{n=F_t}^{L_t} x_{in} - I_{it}^{+} + I_{it}^{-} = d_{it} \qquad i = 1, \ldots, N \quad t = 1, ..., T \qquad (2)$$

$$\sum_{i \in S(k)} r_i x_{in} + st_k z_n^k \le Cap\, y_n^k \qquad k = 1, ..., K \quad n = F_1, ..., L_T \qquad (3)$$

$$z_n^k \ge y_n^k - y_{n-1}^k \qquad k = 1, ..., K \quad n = F_1, ..., L_T \qquad (4)$$

$$\sum_{k=1}^{K} y_n^k = 1 \qquad n = F_1, ..., L_T \qquad (5)$$

$$y_n^k \in \{0,1\} \text{ with } y_0^k = 0 \qquad k = 1, ..., K \quad n = F_1, ..., L_T \qquad (6)$$

$$0 \le z_n^k \le 1 \qquad k = 1, ..., K \quad n = F_1, ..., L_T \qquad (7)$$

$$x_{in} \ge 0 \text{ and integer} \qquad i = 1, ..., N \quad n = F_1, ..., L_T \qquad (8)$$

$$I_{it}^{+} \text{ and } I_{it}^{-} \ge 0 \qquad i = 1, ..., N \quad t = 0, ..., T \qquad (9)$$

The first part of the objective function (1) is a weighted sum of inventory and delay penalties for each period. The second part is the setup penalties, i.e., the alloy changeovers in each sub-period. Thus the objective function seeks a weighted balance between conflicting objectives: stocks, delays and setups. The human schedulers can use their knowledge and experience by varying the values for h_{it}^{-}, h_{it}^{+} and s_k to explore alternative production schedules. Such exploration does not generally give the efficient surface [36], but can provide enough scenarios to guide the decion maker.

Constraints (2) balance inventories, delays, demands and production of items for every item in each period. Constraints (3) not only keep production within the furnace capacity, but also ensure that only items of the same alloy are produced in a particular furnace loading. As y_n^k and y_{n-1}^k are

both binary variables, constraints (4) and the objective (1) force the continuous variables z_n^k to be equal 1 if there is a changeover to alloy *k* or to equal 0 otherwise. Along with constraints (7), the z_n^k variables assume just 0 or 1 values, even if the x_{in} and y_n^k variables are not integer optimal (for example, at nodes during a branch-and-bound search). Constraints (5) and (6) ensure that there is only a single furnace loading in each sub-period.

In captive foundries, lot sizes tend to be large since there are just a few types of items (generally components of standard products) which have a large stable demand, so that the integrality of x_{in} can usually be relaxed. However, in contrast, the integrality condition is necessary for small market-driven foundries with their many small orders. This feature is not taken into account in most lot sizing models [8].

Constraints (9) measure inventory I_{it}^+ and delays I_{it}^- as non-negative variables, but note that in a continuously optimal solution, I_{it}^+ and I_{it}^- will not both be strictly positive, for a given pair (*i*,*t*), due to their positive coefficients in the objective function.

Model (1)-(9) shares some similarities with the GLSP model [8, 24-28]. Sequencing decisions are implicitly determined by the furnace setup variables y_n^k, but differently from the GLSP:

- a sub-period covers the set of products of a given alloy rather than a single product;
- the sub-periods in our model represent the time it takes to process a furnace load, and so have predetermined lengths, whereas in the GLSP the duration of a particular small time bucket is a decision outcome;
- the number of sub-periods per period is fixed, being equal to the number of furnace loads that can be processed per period, although it could be any predetermined number, as in the GLSP.

Thus the GLSP's small and large bucket concepts are both present in the model (1)-(9), since during each period only one type of alloy (an intermediate item which must be used in the period) can be produced together with multiple ordered items. This type of lot sizing and scheduling problems is found in many other applications such as the production of soft drinks or tomato sauce [37].

Depending on the number of items and periods, lot-sizing MIP models are often very large in practice so that even advanced solvers such as Cplex 7.1 [38] are unable to identify probably-optimal solutions in acceptable computational time. Trying to solve the model (1)-(9) with realistic data using MIP solver Cplex 7.1 on a Pentium III 500 MHz with 512 MB of RAM, the default branch-and-cut (B&C) search ran out of memory, achieving only poor solutions.

However, it is generally not worthwhile to invest a lot of computing time in the search for an exact optimal solution, given that input data are often imprecise in small foundries and in

manufacturing in general. A more useful outcome is a quickly-obtained solution of good quality. Delay penalties are usually subjective estimates and the order book is changeable, being updated daily, so that a theoretically optimal solution to model (1)-(9) will almost surely not be the best in practice and should be used as a guide rather than a command. As a result, the exact model may be relaxed to an easier one that includes integer variables only for the first immediate periods (where decisions scarcely change), after which the order book is updated, and the model applied to the next immediate periods, and so on. Such a *rolling horizon* strategy of approaching the problem is widely used in practice. The use of a rolling horizon is not only a useful practical approach (since it takes account of daily changes to the order book), but it works as a very good heuristic strategy to solve a problem even if the order book is not changed, as we will see in the computational experiments where fixed parameters are used.

Several authors have pursued this approach. Clark [39] and Stadtler [40] showed that this flexible approach can handle large multi-level MRP-type problems over long planning horizons with sequence-independent (Stadtler) and sequence-dependent (Clark) setup times. Suerie and Stadtler [41] used the same approach tested on smaller problems with a tight reformulation and valid inequalities providing very good fast solutions. The *relax-and-fix* method as implemented in the current paper fits well into rolling horizon usage, as will be shown below.

4. Rolling Horizon Model

To illustrate how the Rolling Horizon strategy works, suppose each period t is a workday, as in the foundry that motivated this study. Consider a planning horizon of $T = 5$ workdays of which only the first day ($t = 1$) will be scheduled in detail. This is achieved by dividing the first day into $L = h_1 = 10$ sub-periods, as up to L furnace loadings can be processed each day. The remaining days $t = 2,....,5$ have just one sub-period each ($h_2 = h_3 = h_4 = h_5 = 1$). Thus $F_1 = 1$; $L_1 = 10$; $F_2 = L_2 = 11$; $F_3 = L_3 = 12$; $F_4 = L_4 = 13$; $F_5 = L_5 = 14$, i.e., there are $h = 14$ sub-periods n (as illustrated in Figure 2). The variables y_n^k for the larger sub-periods $n = F_2,...,F_5$ are then redefined as "the number of loadings using material k produced in sub-period n".

Only the scheduled decisions relative to the $h_1 = 10$ sub-periods of day 1 are actually implemented. The decisions for the remaining 4 days are used only to evaluate the impact of future available capacity, i.e., to identify a provisional production plan in order to have advance warning of possible production backlogs and be able to act accordingly. Under standard rolling horizon practice, the model is reapplied one period later covering periods $t = 2, \ldots, T+1$ with updated demand data over the rolled-forward T-period horizon, then over periods $t = 3, \ldots, T+2$, and so on, using fresh demand forecasts [42].

To reduce problem complexity and solution time, the integer x_{in} variables are relaxed for sub-periods $n = F_2,...,F_T$, given that these variables' decisions are never in fact implemented. The y_n^k variables for sub-periods $n = F_2,...,L_T$ could also have been relaxed, but initial computational experiments indicated that they should remain integer in order to improve future capacity evaluation.

These modifications result in the following model for rolling horizon use, denominated **RH**:

Model RH:

Minimize $$\sum_{i=1}^{N}\sum_{t=1}^{T}(h_{it}^{-}I_{it}^{-} + h_{it}^{+}I_{it}^{+}) + \sum_{k=1}^{K}\sum_{n=F_1}^{L_1}(s_k z_n^k) \qquad (10)$$

subject to:

$$I_{i,t-1}^{+} - I_{i,t-1}^{-} + \sum_{n=F_t}^{L_t} x_{in} - I_{it}^{+} + I_{it}^{-} = d_{it} \qquad i = 1, \ldots, N \quad t = 1, ..., T \qquad (11)$$

$$\sum_{i \in S(k)} r_i x_{in} + st_k z_n^k \le Cap\, y_n^k \qquad k = 1, ..., K \quad n = F_1, ..., L_1 \qquad (12)$$

$$\sum_{i \in S(k)} r_i x_{in} \le Cap\, y_n^k \qquad k = 1, ..., K \quad n = F_2, ..., L_T \qquad (13)$$

$$z_n^k \ge y_n^k - y_{n-1}^k \qquad k = 1, ..., K \quad n = F_1, ..., L_1 \qquad (14)$$

$$\sum_{k=1}^{K} y_n^k = \frac{L}{h_t} \qquad t = 1, ..., T \quad n = F_t, ..., L_t \qquad (15)$$

$y_n^k \in \{0,1\}$ with $y_0^k = 0$ $\qquad k = 1, ..., K \quad n = F_1, ..., L_1 \qquad (16)$

$y_n^k \ge 0$ and integer $\qquad k = 1, ..., K \quad n = F_2, ..., L_T \qquad (17)$

$0 \le z_n^k \le 1$ $\qquad k = 1, ..., K \quad n = F_1, ..., L_1 \qquad (18)$

$x_{in} \ge 0$ and integer $\qquad i = 1, ..., N \quad n = F_1, ..., L_1 \qquad (19)$

$x_{in} \ge 0$ $\qquad i = 1, ..., N \quad n = F_2, ..., L_T \qquad (20)$

I_{it}^{+} and $I_{it}^{-} \ge 0$ $\qquad i = 1, ..., N \quad t = 1, ..., T \qquad (21)$

Constraints (3) are now replaced by (12) for the first period and (13) for the remaining periods. Constraint (5) is replaced by (15) which imposes exactly $L/?_t$ setups in sub-period n, i.e., the number of loads in period t divided by the number of sub-periods in period t. For example, period 1 has 10 sub-periods and can handle 10 loads, so $L/?_1 = 10/10 = 1$ (and $n = F_1,\ldots,L_1$, i.e., $n = 1,\ldots,10$), whereas period 2 has one sub-period and can handle 10 loads, so $L/?_2 = 10/1 = 10$ (and $n = F_2,\ldots,L_2$, i.e., $n = 11$).

Model RH maintains the similarity to the GLSP, adapting its small-bucket/large-bucket concepts for rolling horizon use. The large-bucket time period used for scheduling a whole day, for example, is split into several small-bucket scheduling time periods (for instance, 10 loadings of materials). Days 2 to 5 are, temporarily, indivisible large buckets with production of multiple materials.

5 Solution Methods

Model RH is much smaller than the model (1)-(9), but still not small enough to be solved optimally with realistic data using the MIP solver Cplex 7.1 within acceptable computing time. However, it is possible to sub-optimally solve the model using the *relax-and-fix* method. This involves the sequential solution of a series of partially relaxed MIPs, one per period, each one with a small enough number of integer variables to be quickly solved. As the series progresses in time from the first period to the last, each set of integer variables are permanently fixed at their solution values. The relax-and-fix procedure solves the model RH in two steps, as follows:

1. *Relax* all integer variables, except the first day's binary variables y_n^k ($n = F_1, ..., L_1$), representing furnace loadings in period 1 and being the most important decisions in the rolling horizon method. Solve this relaxed problem.
2. *Fix* the first day's y_n^k ($n = F_1, ..., L_1$) variables at their binary values from the solution in step 1. The y_n^k ($n = F_2, ..., L_T$) variables and x_{in} ($n = F_1, ..., L_1$) variables are specified as integer. Solve this partially fixed problem.

The problem in step 1 is solved using one of the four methods (RF, DH, DN and SA) described in sections 5.1 to 5.4 below. The problem in step 2 can be optimally solved in a few seconds with the Cplex MIP solver, since a binary variable y_n^k which is fixed to 1 implies, by constraints (12) and (15), that $x_{in} = 0$ for all $i \notin S(k)$, i.e., products that do not use material k are not manufactured in sub-period n, thus eliminating many integer variables and constraints. Consequently, the solution methods developed in the rest of this paper focus on step 1.

5.1 Basic Relax-and-Fix Method (RF):

The basic approach to solving step 1 of the *relax-and-fix* method simply uses the incumbent solution that results from running the Cplex MIP solver for 3, 6 and 12 minutes respectively for small, medium and large problems (as defined in Table 1 of section 4.1). This method is denoted RF.

5.2 Descent Heuristic (DH)

The basic RF method is dependent on a MIP solver for both MIP problems in the two steps that arise in the relax and fix procedure above. The first problem is to solve (and then fix) just the first day's binary variables. The second problem, in step 2, is to try to find an optimal solution for variables y_n^k ($n = F_2, ..., L_T$) and x_{in} ($n = F_1, ..., L_1$).

To solve the first MIP, a local search *descent heuristic* (DH) [43-45] can be used to find good values for period 1's binary y_n^k ($n = F_1, ..., L_1$) variables. Starting, for example, with a random solution and fixing these variables, all the other integer variables are relaxed and the resulting linear programming model is solved. In the next local search iteration, the period 1 binary variables are modified and the linear program is solved again to obtain a *neighbouring solution*. Depending on certain criteria, the neighbouring solution may become the current solution. The local search then proceeds to the next iteration. The best solution encountered as the search progresses is recorded. When the stopping criterion of the local search holds, the y_n^k ($n = F_1, ..., L_1$) variables are fixed at the best solution found.

In order to implement the DH method (Algorithm 1), it is necessary to define a series of parameters. In this paper the solution representation in the descent heuristic consists of a h_1-vector of integers, $\boldsymbol{v} = (v_1, ..., v_{h_1})$, where v_n contains the type of material scheduled for sub-period n in the first day, that is, $v_n = k$ if and only if $y_n^k = 1$. For example, when h_1=10, the solution vector $\boldsymbol{v}$ = (2, 2, 20, 1, 4, 4, 8, 2, 10, 3) means that material type 2 is made in the first two sub-periods, material type 20 in the third sub-period, and so on.

Three alternative ways of obtaining a starting solution for the descent heuristic were at first considered:

1. For $n = 1,...,h_1$, choose the value of v_n to be k with the probability given by $|S(k)| / N$ where $|S(k)|$ is the size (cardinality) of the set $S(k)$. Thus, the more products that can be made from k, the more likely it is that k will be selected;
2. Run a MIP solver for a few minutes to obtain an initial heuristic solution;
3. For $n = 1,...,h_1$, choose the value of v_n to be k, uniformly sampled from $\{1, ..., K\}$.

However, after initial tests the first way was selected and the other two were discarded.

The search stops after 1000 iterations, which was found to be sufficient to obtain a good solution in acceptable computing time. This number of iterations was fixed as the stopping criterion in order to fairly compare the three types of heuristic.

In order to determine neighbouring solutions it is necessary to specify how the vector $\boldsymbol{v}$ is changed, which defines decisions to the first h_1 sub-periods. Just one sub-period has its value

changed (in the Diminishing Neighbourhood Heuristic more than one can be changed in the beginnig, but ends with just one changing). The neighbourhood move implemented slightly biases the selection towards more widely-used alloys, and was adopted after initial testing showed its positive impact. Two alternative procedures were used to randomly choose n^*, the sub-period to change the material k:

1. With 90% probability: The value of n^* is uniformly sampled from the set $\{1,...,h_1\}$.
2. With 10% probability: Let $k = v_n$ be the material currently produced in a given sub-period n. We want that the more products $S(k)$ made from material k, the less the chance of selecting sub-period n. So, sample the value of k with probability $(N-|S(k)|) / (N(K^*-1))$ where K^* is the number of different materials to be produced in period 1. The value of n^* is then uniformly sampled from those sub-periods in which material k is produced.

Once n^* is chosen, a new key material k is selected in one of two ways:

1. With 90% probability: k is uniformly randomly sampled from the set $\{1,...,K\}$,
2. With 10% probability: k is sampled from the set $\{1,...,K\}$ with probability $|S(k)| / N$, i.e., the more products $S(k)$ that can be made from k, the greater the likelihood of selecting k.

Algorithm 1 (Descent Heuristic Procedure)

1. Select a starting solution: $\boldsymbol{v} = (v_1, ..., n_{h_1})$, which means $\{ y_n^k \mid (n = F_1, ..., L_1) \}$.
2. Relax the integer variables x_{in} $(n = F_1, ..., L_1)$ and y_n^k $(n = F_2, ..., L_T)$ as explained before and solve the LP problem resulting.
3. Record as the incumbent solution that one obtained from the linear programme in step (2).
4. Repeat steps (4.1) to (4.3) for 1,000 iterations:
   4.1. Generate a neighbouring solution of the incumbent: select a sub-period n in period 1 ($F_1 \le n \le L_1$) and a new alloy k, as described above; let $v_n = k$.
   4.2. Fix the decisions on the first period; that is, if $v_n = k$ then $y_n^k = 1$ and $y_j^k = 0$ for $j \neq k$; solve the relaxed linear programme.
   4.3. If the neighbouring solution provides a better objetive function value in (10) than the incumbent solution then it becomes the incumbent.
5. Fix the values of the y_n^k $(n = F_1, ..., L_1)$ variables the incumbent solution. Restore the x_{in} $(n = F_1, ..., L_1)$ and the y_n^k $(n = F_2, ..., L_T)$ to be integer variables, and solve the resulting small MIP to obtain an optimal integer solution.

Researchers have proposed many ways to improve the descent heuristics performance, including [43]-[51]. In this paper, we use two strategies: Diminishing Neighbourhood search

method and Simulated Annealing, with the same basic parameters as the local search described above.

5.3 Diminishing Neighbourhood (DN) Search

This method adapts the local search described in the previous section, beginning with a large neighbourhood to encourage diversity and then gradually diminishing its size so as to increasingly intensify the search. Too small a neighbourhood could cause premature convergence and increase the risk of stagnation at a local optimum, while too large a neighbourhood would lead to random meandering and an inefficient search. The search starts with the largest possible neighbourhood, i.e., all the first day's h_1 variables y_n^k (i.e., n_n) in a solution can be changed in step 4.1 of the descent heuristic procedure. After a given number of iterations the neighbourhood size is reduced, i.e., only h_1-1 randomly uniformly selected variables y_n^k (n_n) in a solution can be changed. During the search, neighbourhood size is repeatedly diminished. The search ends with a neighbourhood where just one position is changed, i.e., as in the descent heuristic in section 5.2.

For each size Z = 1,…,10 of neighbourhood, 18(11-Z)+1 iterations are carried out at step 4, summing to a total of 1000 iterations over the whole search. Thus 19 iterations are carried out when Z = 10 at the start of the search, 37 when Z = 9, and so on, increasing to 181 iterations when Z = 1 at the end of the search. Clark [52] successfully used a similar method for lot-sizing on a drinks canning line.

5.4 Simulated Annealing (SA)

Simulated Annealing is a variant of the local search descent heuristic that tries to avoid getting trapped at a local optimum by permitting worsening moves away with probability:

$$p(\Delta ofv) = e^{-\left(\frac{\Delta ofv}{Temp}\right)} \qquad (22)$$

where *Temp* is a gradually-cooling "temperature" and ? *ofv* the amount by which the new move worsens the objective function value. As the search progresses, the best solution encountered is recorded.

Previous computational tests indicated that the following parameters produce, in general, the best results. The starting temperature $Temp_{start}$ is a function of the initial solution ([48]):

$$Temp_{start} = \frac{m \times \text{Value of the Initial Solution}}{-\log(q)} \qquad (23)$$

where m = 0.6 and q = 0.9 indicate that a solution which is 60% worse than the current one has 90% probability of acceptance at the start of the search. 50 iterations were allowed in order to reach equilibrium at a given temperature before cooling but only 10 iterations after a solution was accepted, even when worse. Each time the temperature was cooled in this way, it was reduced by 5%. In addition, each time a worse solution was accepted, the temperature was again cooled (slightly) as follows:

$$Temp_{new} = Temp_{old} - \left(0.1 \times Temp_{old} \times \frac{\Delta ofv}{ofv(S)} \right) \qquad (24)$$

where *ofv*(S) is the objective function value for the previous solution. The worse the accepted solution, the greater the reduction in temperature, thus making the acceptance of future worse solutions less likely from then on.

6. Computational Experiments

The computational experiments were divided into two parts. In order to evaluate the method in different situations, the first part used randomly generated data based on modified intervals based on [25]. The second part made use of an order book from a real-world foundry, so that the method's outcome could be compared with schedules used in practice.

Before describing the data generation, consider following parameter definitions:

a_i the number of days by which item *i* is already delayed at the beginning of the schedule, i.e., at t = 0.

r_i weight of item *i* (previously defined in section 3)

A value a_i = 0 means that the item *i* due date is day 1, and a_i < 0 means that item *i* is not delayed when the planning begins. Suppose that an item *i* is already delayed by a_i > 0 at the beginning of the planning. Then, at the end of period *t*, t = 1,…,T the item's delay will be a_i+t. Its delay penalty h_{it}^- is calculated as $r_i(a_i+t)$ so as to increasingly penalise any further delay in its production. However, if item *i* is not already delayed, i.e., $a_i \leq 0$, then it can be produced up to period $1+|a_i|$ without delaying. Thus, for $t = 1+|a_i|$, a positive value $I_{it}^- > 0$ means one day of delay and so h_{it}^- is also calculated as $r_i(a_i+t)$ from this period onwards.

Furthermore, if item *i* is not delayed at the beginning of the schedule, then the variable I_{it}^+ can be positive (i.e., item *i* can be produced before its due date) and its inventory penalty h_{it}^+ is defined as proportional to its weight r_i. In this case, to force I_{it}^- to be zero (given that there is no

delay), the delay penalty h_{it}^- is set to be a very large number G. Similarly, in the case of a delay, its inventory penalty h_{it}^+ is also set to G to force I_{it}^+ to be zero.

In summary:

if $\alpha_i \geq 0$ (i.e., item i is already delayed at the start, or the due date is day 1) then

for t=1,...,T, let $h_{it}^- = r_i(a_i+t)$ and $h_{it}^+ = G$;

else (i.e., ($\alpha_i < 0$, meaning that item i will be delayed after period $1+|\alpha_i|$)

for t=1,...,$|a_i|$, let $h_{it}^- = G$ and $h_{it}^+ = r_i$;

for t= $1+|a_i|$,...,T, let $h_{it}^- = r_i(a_i+t)$ and $h_{it}^+ = G$.

If orders for an item have different due dates, then this item will be considered as two distinct items in order to have two different delay penalty values. Although this could considerably increase the model size, in practice it will do so only a little at most, since in small market-driven foundries only a few items are doubled ordered within the one-week planning horizon. Note that this doubling of items will not create false setups (as it would in classical lot sizing models with item setups) since the doubled items belong to the same set $S(k)$, as in constraints (12).

It might be tempting to explicitly prohibit delays in the model, but as these are frequently unavoidable, such a ban would result in infeasible problems and would be unrealistically rigid. It is realistic to include the possibility of delays in the model and let the human scheduler manage them, for example, by calibrating a production priority parameter for each individual items (b_i) that can be included in the computation of h_{it}^-. Moreover, delay variables help the scheduler to evaluate due dates. For instance, if d_{i1} = 30, then a model solution I_{i1}^- = 10, $I_{i,2}^-$ = 5, $I_{i,3}^-$ = 0 flags that the demand for item i will be fully met only after 2 days of delay. In this case, the client could be alerted and, if necessary, the scheduler could increase the value of parameter b_i for that item or renegotiate the item's due-date.

6.1 Generation of Test Data

Previous experience [42] indicates that certain parameters may affect solution quality, namely:

- problem size (N,K)
- size of setup penalty st_k
- tightness of capacity Cap

Larger problems, bigger setup penalties, and tighter capacity are each expected *a priori* to adversely affect solution quality and computing time, but may do so in different degrees for each solution method. The uniform sampling intervals used to randomly generate the test data were based on those in [25] and are shown in Table 1. Though not encompassing every possible

situation, the values are sufficiently typical to be confident that the test results will point to generally applicable conclusions.

The furnace capacity was generated as follows: first calculate the resources needed to exactly produce the total item demand over the planning horizon (in this case 5 periods, i.e., 50 furnace loadings); then add the total setup time needed if the furnaces were setup just once for each alloy; finally divide by the number of furnace loads, i.e., 50. Hence:

$$C = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} d_{it} r_i + \sum_{k=1}^{K} st_k}{50} \qquad (25)$$

Thus in Table 1 four different levels of the tightness of Furnace Capacity *Cap* are shown (i) Very Loose capacity: *Cap* = *C* / 0.6; (ii) Moderately Loose: *Cap* = *C* / 0.8; (iii) Moderately Tight: *Cap* = *C* / 1.0; (iv) Very Tight: *Cap* = *C* / 1.2.

The parameters (*N*, *K*), s_k and *Cap* were varied in a 3-factor experimental design. Each factorial combination was generated 10 times, using a different random seed each time, resulting in a total of 3×2×4×10 = 240 instances.

6.2 Solution Quality

In this section, we first analyse the quality of the solutions obtained by the relax-and-fix (RF) method described in section 5.1. We then compare the results from the three neighbourhood search approaches and the basic RF method. Finally we consider computing times.

6.2.1 Evaluation of the RF method

In order to evaluate the performance of the RF method, we used the solutions and lower bounds obtained by the Cplex 7.1 solver applied to model (1)-(9) in a general purpose branch-and-cut search.

Note that a solution to model RH is not a solution to model (1)-(9), as just the first day's loadings are scheduled and actually implemented, whereas the other days are planned only approximately. However, the application *T* times of model RH, starting consecutively at periods 1, 2, …, 5, with an always-shortening horizon (*T* = 5,4,3,2,1), will provide a feasible solution to model (1)-(9), enabling a comparison of results, similar to the internally rolling schedule in Stadtler [40]. The final value of the objective function is gradually accumulated over the *T* applications of model RH, each of which contributes its period 1 part:

$$\sum_{i=1}^{N} (h_{i1}^{-} I_{i1}^{-} + h_{i1}^{+} I_{i1}^{+}) \qquad (26)$$

of expression (10), i.e., excluding the part for periods 2 onwards: $\sum_{i=1}^{N}\sum_{t=2}^{T}(h_{it}^{-}I_{it}^{-}+h_{it}^{+}I_{it}^{+})$.

Table 2 shows the variation of the mean objective value compared to the lower bounds supplied by Cplex after running for one hour. The variation is calculated as:

$$\text{Variation} = \frac{\text{Method Solution - Cplex Lower Bound}}{\text{Cplex Lower Bound}} \times 100\% \qquad (27)$$

where the method is either Cplex or RF.

As noted in section 3, attempts at solving model (1)-(9) to simultaneously schedule all furnace loadings over the whole 5-day horizon resulted in mediocre solutions. The Cplex incumbent solution, after 1 hour, was on average 22.26% worst than the lower bound. Even if more time was allowed (10 hours for some instances) the Cplex solutions were still poor. On the other hand, running the RF method for 3, 6, and 12 minutes for the small, medium and large problems respectively resulted in better solutions for all three cases that were on average 8.6% worse than the Cplex lower bound. Irrespective of the method, Table 2 shows that, while this gap tends to grow with problem size, there is no clear relationship with tightness of capacity or size of setup penalty.

6.2.2 Evaluation of the DH, DN and SA methods

After evaluating the quality of the RF method, we now compare the three neighbourhood search methods (DH, DN and SA) and the basic RF method using model RH.

The basic RF method of section 5.1 used the MIP incumbent solution found within the time limits. Table 3 shows the percentage of instances for which this method found the MIP optimal solution. Observe that for small problems (N = 10, K = 2) the method easily found optimal solutions to all of the MIPs, but the percentage of optimal solutions found reduced as problem size increased. Again, there is no clear relationship with tightness of capacity or size of setup penalty.

Table 4 shows the variation of the mean objective value for the *DH*, *DN* and *SA* local search heuristics compared to the basic RF method, calculated as:

$$\text{Variation} = \frac{\text{Heuristic Solution - RF Solution}}{\text{RF Solution}} \times 100\% \qquad (28)$$

Overall the three heuristics performed nearly as well as the basic RF method, the best being SA, followed by DN. The DH heuristic converged rapidly to a local optimum while the DN and SA ones took many more iterations to achieve their best solution.

Note in Table 4 that the performance of the DH, DN and SA heuristic relative to the basic RF method improved as problem size increased. For small and medium problems, the DH method had

the worst mean performance (4.36% and 3.38%, respectively), followed by DN (1.93% and 2.72%), then SA (1.02% and 1.48%). For large problems, the mean performance of DH improved to 0.90% and in fact was better than DN (1.31%), though worse than SA (0.56%). This difference in relative performance might be explained by two reasons. First, for problems of all sizes, both DH and SA will initially follow the same search trajectory, but when either reaches a local optimum, DH has no way to jump out of it and so its search stagnates there, whereas SA can go to a worse solution to get away from the local optimum. The DN search follows a very different path than DH and SA. If it gets stuck at a local optimum, this tends to be when its neighborhood is near its minimum size towards the end of the search. Secondly, recall from Table 3 that the larger the problem, the fewer the MIPs that Cplex 7.1 is able to solve to optimality, thus weakening the basic RF method. This means that the DH, DN and SA methods are being compared against probably-suboptimal solutions obtainable within the branch-&-cut search time limits.

Table 3 also shows that tightness of capacity does not much affect the basic RF method, with the possible exception of medium problems. However, Table 4 indicates that, regardless of problem size, there is generally less variation (i.e., better performance) of the DH, DN and SA methods relative to RF when capacity is tight and/or when setup penalties are small.

The computing time spent by the DH, DN and SA methods to solve each MIP was about 1, 3 and 5 minutes respectively for small, medium and large problems. These are viable times for practical use and faster than the time spent by the basic RF method (3, 6 and 12 minutes respectively). The solution of the final MIP in step 5 (algorithm 1) of the DH, DN and SA methods was limited to a maximum of 5 minutes of computing time. In practice, the MIP was usually solved by Cplex in less than 10 seconds, even for large problems, because the binary y_n^k ($n=F_1, \ldots, L_1$), variables had been previously fixed, leaving only the non-zero integer x_{in} values to be optimised.

6.3 Evaluation in a Small Foundry

The methods were also tested on real world instances at a small foundry that used a 5-day planning horizon, with 10 furnace loadings per day, totalling 50 over the whole horizon. The furnace had a capacity of 360kg per load, each one taking approximately 2 hours. At the time of testing, the prevalent situation at the foundry was that many items were delivered with delays, some of them up to 100 days.

This is clearly a complex scheduling challenge but, like other small companies, the foundry has neither the resources nor the sophistication to invest in scheduling research. There are constant delays in deliveries, clients are frequently lost, utilization of equipment and manpower is inefficient, leading to queues of moulds waiting for alloys. At other times, excessive quantities of alloys are produced which have to be recycled as scrap materials.

The methods were tested with an order book from the foundry. Just a single order book was used due to the difficulties of obtaining stable static data and the difficulties of posterior comparison of results with the schedules used in practice.

The solutions obtained by the DH, DN and SA method were almost identical. They were compared with the foundry's own manual schedule for an order book of 383 product types requiring 19 different alloys. Order weights varied a great deal (from 0.5 kg to 200 kg), as did their quantities (from 1 to 1,000 items). The initial stocks and backlogs were respectively 0 and 526,818 item-days (corresponding to 0 and 426,528 kg-days), where item-days (kg-days) are calculated by multiplying the quantity (weight) by the number of days in stock or backlogged. Tables 5 and 6 show the 5-day schedules output by the DH method compared to the foundry's manually produced schedules. The "Day 5" line at the end of the DH method's schedule shows that the final stocks and backlogs were respectively 0 and 3476 item-days (0 and 48,195 kg-days). On the other hand, the foundry's manual schedule resulted in final stocks and backlogs of 3 and 23,237 item-days respectively (210 and 81,500 kg-days). In other words, the DH schedule reduced item-day delays by 85% and kg-day delays by 40%, a very substantial gain in efficiency.

The DH schedule used less capacity (93.1% usage) than the foundry's manual schedule (98.7%) and produced a wider variety of types (34 alloys in opposition to 28). Note from Tables 5 and 6 that the 5-day totals of the DH backlogs and stocks are substantially lower, i.e., respectively 27% and 90% by weight, and 15% and 71% by number of items. This reflects the foundry's concern to maximize utilization of capacity, a policy which tends to prioritize of larger lots, resulting in the repeated postponement or early production of orders, thus creating larger backlogs and inventories as can be seen in Tables 5 and 6.

Excluding time for input data, the DH method generated a schedule in ten minutes, compared to the two days (16 working hours) of elapsed time that it took to specify the manual schedule.

This improved scheduling permits better negotiation of delivery dates (with simulation, for example), reducing promises of impossible deadlines. It also avoids an additional problem that frequently occurs in the foundry (and which the proposed method easily resolves), namely the over-utilization of the furnaces in an attempt to reduce delays. Such overloading is physically possible but unadvisable as smelting takes longer, the furnace internal coating deteriorates sooner, and more energy is consumed, thus increasing production costs.

A further advantage for the foundry is that, when the order book is updated daily, rescheduling of the following days is easily carried out, allowing the inclusion of new orders.

7 Conclusions

In this paper a mixed integer linear programme (MIP) was proposed to model the production planning and scheduling in small foundries. The model integrated lot sizing and scheduling on a single capacitated machine (furnace) in a production environment where key alloys are first produced and subsequently transformed into a number of ordered items made from just one type of alloy. The foundry's main concern was to minimize delays. The inventory of alloys was forbidden, and at any time only one type of alloy could be produced. The setups for changing alloys were sequence-independent. There were neither setup times nor costs between the production of final products. Lot sizes of final items were assumed to be integers and backorders were allowed.

It was not possible to optimally solve the overall model within viable computing time, even using an advanced heavy-duty MIP solver. In order to efficiently but approximately solve the model, a rolling horizon approach and associated relax-and-fix procedure was developed. Four solution methods were proposed using a basic relax-and-fix (RF) approach and three variants of neighbourhood search. These four methods were tested with 240 generated instances based on Haase and Kimms (2000), and showed that the relax-and-fix approach provides a good compromise between speed and quality of solution, that local search is faster with slightly worse solutions, and that, for the data used, simulated annealing generally resulted in better solutions than the other two local search variants.

Tests were also carried out with a real-world instance from a foundry. The results showed that not only do the methods help small foundries to considerably reduce delays, but also that the improved schedules are generated in a very small fraction of the time of those created manually in the foundry. Ongoing efforts are continuing with the foundry to obtain better quality data with the aim of making a more precise comparison between the schedules currently used in practice and those output by the proposed method. In parallel, other small foundries in the region are being sought for further case comparisons. To facilitate such collaboration, a tool with a visual interface is being developed to output schedules and alloy mixes, within a wider objective of providing production planning and scheduling software for small and medium-sized foundries.

* *Acknowledgements*: The authors would like to thank the referees for their valuable reviews and helpful suggestions, and the National Research Council (CNPq), Brazil, for financial support.

References

1. ABIFA - Associação Brasileira de Fundição. *Relatório anual do setor de fundição*. http://www.abifa.org.br, 2005.
2. Billington PJ, McClain JO, Thomas LJ. Mathematical programming approaches to capacity mrp systems: Review formulation and problem reduction. *Management Science*. 1983; **29** (10): 1126-1141.
3. Bahl HC, Ritzman LP, Gupta JND. Determining lot sizes and resource requirements: A review. *Operations Research.* 1987; **35**: 329-345.
4. Maes J, van Wassenhove LN. Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *Journal of Operational Research Society*. 1988; **39** (11): 991-1004.
5. Goyal SK, Gunasekaran A. Multi-stage production-inventory systems. *European Journal of Operational Research*. 1990; **46**: 1-20.
6. Potts CN, Van Wassenhove LN. Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *Journal of Operational Research Society*. 1992; **43** (5): 395-406.
7. Kuik R, Salomom M, Van Wassenhose LN. Batching decisions: Structure and models. *European Journal of Operational Research.* 1994; **75**: 243-263.
8. Drexl A and Kimms. A Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research.* 1997; **99**: 221-235.
9. Potts CN, Kovalyov MY. Scheduling with batching: A review. *European Journal of Operational Research.* 2000; **120**: 228-249.
10. Karimi B, Fatemi Ghomi SMT, Wilson JM. The capacitated lot sizing problem: A review of models and algorithms. *Omega.* 2003; **31**: 365-378.
11. Santos-Meza E, Santos MO, Arenales MN. Lot-sizing problem in an automated foundry. *European Journal of Operational Research.* 2002; **139** (2): 490-500.
12. Araujo SA, Arenales MN. Planejamento e programação da produção numa fundição cativa automatizada de grande porte. *Investigação Operacional.* 2004; **24**: 197-219.
13. Sounderpandian J, Balashanmugam B. Multiproduct multifacility scheduling using the transportation model: A case study. *Production and Inventory Management Journal.* 1991; **32** (4): 69-73.
14. Gravel M, Price WL, Gagné C. Schedulings jobs in an alcan aluminium foundry using a genetic algorithm *International Journal of Production Research.* 2000; **38** (13): 3031-3041.
15. Tang L, Liu J, Rong A, Yang Z. A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*. 2001; **133**: 1-20.

16. Petersen CM, Sorensen KL, Vidal RVV. Inter-process syncronization in the steel production. *International Journal of Production Research.* 1992; **30** (6): 1415-1425.
17. Hamada K, Baba T, Sato K, Yufu M. Hybridizing a genetic algorithm with rule-based reasoning for production planning. *IEEE Expert.* 1995; **10**: 60-67.
18. Bowers MR, Kaplan LA, Hooker TL. A two-phase model for planning the production of aluminum ingots. *European Journal of Operational Research.* 1995; **81**: 105-114.
19. Hendry LC, Fok KK, Shek KW. A cutting stock and scheduling problem in the copper industry. *Journal of Operational Research Society*. 1996; **47** (1): 38-47.
20. Lee HS, Murthy SS, Haider SW, Morse DV. Primary production scheduling at steelmaking industries. *IBM Journal of Research and Development.* 1996; **40** (2): 231-252.
21. Lopes L, Carter MW, Gendreau M. The hot strip mill production scheduling problem: A tabu search approach. *European Journal of Operational Research.* 1998; **106**: 317-335.
22. Tang L, Liu J, Rong A, Yang Z. A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research,* 2000; **120**: 423-435.
23. Tang L, Liu J, Rong A, Yang Z. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan iron & steel complex. *European Journal of Operational Research.* 2000; **124**: 267-282.
24. Haase K. Capacitated Lot-Sizing with Sequence Dependent Setup Costs. *Operational Research Spektrum*. 1996; **18**: 51-59.
25. Haase K. and Kimms A. Lot Sizing and Scheduling with Sequence Dependent Setup Costs and Times and Efficient Rescheduling Opportunities. *International Journal of Production Economics*. 2000; **66**: 159-169.
26. Fleischmann B. and Meyr H. The General Lotsizing and Scheduling Problem. *Operational Research Spektrum*. 1997; **19**: 11-21.
27. Meyr, H. Simultaneous Lotsizing and Scheduling by Combining Local Search with Dual Reoptimization. *European Journal of Operational Research*. 2000; **120**: 311-326.
28. Meyr, H. Simultaneous Lotsizing and Scheduling on Parallel Machines. *European Journal of Operational Research*. 2001; **139** (2): 277-292.
29. Gupta, D. and Magnusson T. The Capacitated Lot-Sizing and Scheduling Problem with Sequence-Dependent Setup Costs and Setup Times. *Computers and Operations Research*. 2005; **32**: 727-747.
30. Dillenberger C, Escudero LF, Wollensak A and Zhang W. On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*. 1994; **75**: 275-286.

31. Beraldia, P, Gianpaolo G, Emanuela G, and Antonio G. Scenario-based planning for lot-sizing and scheduling with uncertain processing times. *International Journal of Production Economics*. 2006; in press.
32. Wolsey LA. *Integer programming*. New York: Wiley, 1988
33. Kelly, JD and Mann JL. Flowsheet Decomposition Heuristic For Scheduling: a Relax-and-Fix Method. *Computers and Chemical Engineering*. 2004; **28**: 2193-2200.
34. Kuik R, van Wassenhose LN and Maes J. Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lotsizing in Bottleneck Assembly Systems. *IIE Transactions*. 1993; **25**, 1: 62-72.
35. Teghem, J, Pirlot M, and Antoniadis C. Embedding of Linear Programming in a Simulated Annealing Algorithm for Solving a Mixed Integer Production Planning Problem. *Journal of Computational and Applied Mathematics*. 1995; **64**, 91-102.
36. Miettinen KM. *Nonlinear multiobjective optimization.* Kluwer, 1998.
37. Araujo SA, Arenales MN, Clark AR. Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. *Congreso Latino-Americano de Investigacion Operacional (CLAIO*). Havana, 2004.
38. ILOG Cplex 7.1 User's manual. ILOG SA BP 85 9 Rue de Verdun 94253. Gentilly, France. http://www.ilog.com, 2001.
39. Clark AR. Optimization Approximations for Capacity Constrained Material Requirements Planning Problems. *International Journal of Production Economics.* 2003; **84**: 115-131.
40. Stadtler, H. Multilevel Lot Sizing with Setup Times and Multiple Constrained Resources: Internally Rolling Schedules with Lot-Sizing Windows. *Operations Research*. 2003; **51**: 487-502.
41. Suerie, C and Stadtler H. The Capacitated Lot-Sizing Problem with Linked Lot-Sizes. *Management Science*. 2003; **49**: 1039-1054.
42. Clark, AR. Rolling Horizon Heuristics for Production and Setup Planning with Backlogs and Error-Prone Demand Forecasts. *Production Planning and Control*. 2005; **16**: 81-97.
43. Reeves CR. *Modern heuristic techniques for combinatorial problems.* Blackwell, 1993.
44. Pirlot M. General Local Search Methods. *European Journal of Operational Research*. 1996; **92**: 493-511.
45. Aarts EHL, Lenstra JK (Editors). *Local search in combinatorial optimization.* Chichester: Wiley, 1997.
46. Goldberg D E. *Genetic algorithms in search optimization and machine learning*. Massachusetts: Addison Wesley, 1989.
47. Laguna M. Tabu search tutorial. *II Escuela de Verano Latino-Americana de Investigation Operativa*. Rio de Janeiro, 1995.

48. Diaz A, Glover F, Ghaziri HM, González JL, Laguna M, Moscato P, Tseng FT. *Optimización heurística y redes neuronales*. Spain: Editorial Paraninfo, 1996.
49. Gen M, Cheng R. *Genetic algorithms & engineering Design.* Wiley, 1997.
50. Glover F, Laguna M. *Tabu search*. Norwell, Massachusetts: Kluwer, 1997.
51. Glover, F. W., and G. A. Kochenberger (eds.). (2003). "Handbook of Metaheuristics". Boston: Kluwer.
52. Clark AR. Hybrid heuristics for planning lot sizes and setups. *Computers and Industrial Engineering*. 2003; **45** (4): 545-562.

Figure and table captions

Figure 1. Main activities in the foundry

Figure 2: Periods and sub-periods in a rolling horizon strategy.

Table 1: Parameters used for generation of uniformly-distributed test data.

Table 2: Mean solution variation (%) of the solver Cplex and RF methods compared to the Cplex Lower Bounds.

Table 3: Percentage of test problems in which Cplex found an optimal solution for all the *RF* MIPs

Table 4: Mean solution variation (%) of the DH/DN/SA heuristics compared to the basic RF method.

Table 5: DH solution compared to the schedule practiced in a foundry (in item-days)

Table 6: DH solution compared to the schedule practiced in a foundry (in kg-days)

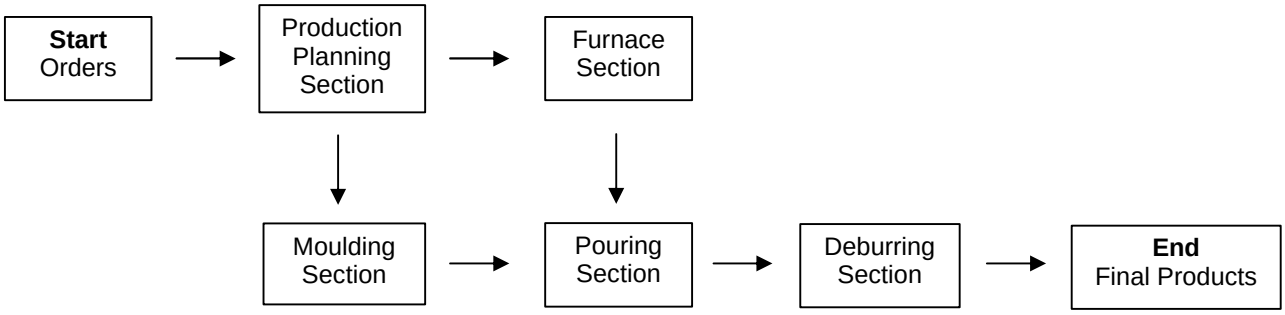

Figure 1. Main activities in the foundry

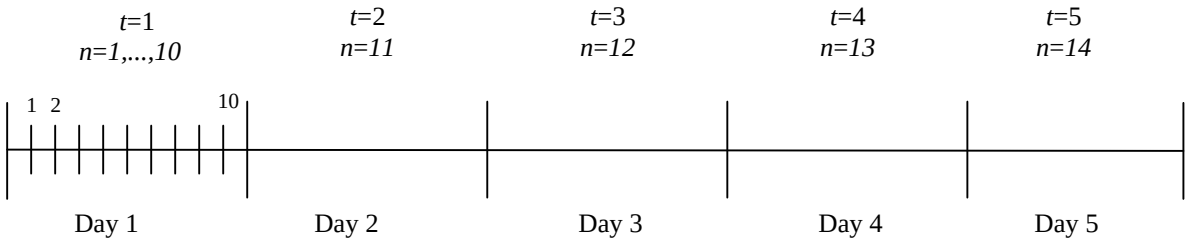

Figure 2: Periods and sub-periods in a rolling horizon strategy

Table 1: Parameters used for generation of uniformly-distributed test data.

Parameters	**Values**
Number of Items and Alloys: (N, K) pairs	Small Problem: (10, 2) Medium Problem: (50, 10) Large Problem: (100, 20)
Number of Days:	5
Demand: d_{it}	[10, 60]
Days of Delay: a_i	[-10, 10]
Physical Weight of Item: r_i	[1, 30]
Setup Time of Alloy: st_k	[5, 10]
Setup Penalty of Alloy: s_k	Low: $5 \times st_k$ High: $50 \times st_k$
Tightness of Furnace Capacity: *Cap*	C / 0.6, C / 0.8, C / 1.0, C / 1.2

Table 2: Mean solution variation (%) of the solver Cplex and RF methods compared to the Cplex Lower Bounds.

Method:		Cplex			RF		
Setup Penalty Factor s_k		$5\times st_k$	$50\times st_k$	Mean	$5\times st_k$	$50\times st_k$	Mean
Problem Size	Capacity						
Small: (N,K) = (10, 2)	C/0.6	6.94	5.37	6.16	1.96	0.06	1.01
	C/0.8	4.20	4.56	4.38	1.82	1.29	1.55
	C/1.0	5.19	5.39	5.29	2.31	3.18	2.74
	C/1.2	10.93	8.46	9.70	2.80	3.60	3.20
	Mean	6.82	5.95	6.38	2.22	2.03	2.13
Medium: (N,K) = (50, 10)	C/0.6	26.83	41.26	34.04	10.35	19.91	15.13
	C/0.8	19.39	31.52	25.45	6.33	14.85	10.59
	C/1.0	25.65	31.64	28.65	6.62	11.98	9.30
	C/1.2	26.72	27.88	27.30	5.74	10.15	7.94
	Mean	24.65	33.07	28.86	7.26	14.22	10.74
Large: (N,K) = (100, 20)	C/0.6	35.79	44.62	40.21	11.69	27.34	19.51
	C/0.8	31.26	34.76	33.01	9.19	16.32	12.76
	C/1.0	27.79	31.38	29.58	8.01	13.57	10.79
	C/1.2	21.46	25.31	23.39	6.67	10.62	8.64
	Mean	29.08	34.02	31.55	8.89	16.96	12.93
Overall Mean (%)		20.18	24.35	22.26	6.12	11.07	8.60

Table 3: Percentage of test problems in which Cplex found an optimal solution for all the *RF* MIPs

Problem Size	Small: (N, K) = (10, 2)				Medium: (N, K) = (50, 10)				Large: (N, K) = (100, 20)			
Setup Penalty s_k	C/0.6	C/0.8	C/1.0	C/1.2	C/0.6	C/0.8	C/1.0	C/1.2	C/0.6	C/0.8	C/1.0	C/1.2
$5 \times st_k$	100	100	100	100	62	66	68	80	44	42	36	56
$50 \times st_k$	100	100	100	100	68	68	86	90	44	42	32	44

Table 4: Mean solution variation (%) of the DH/DN/SA heuristics compared to the basic RF method.

Method:		DH			DN			SA		
Setup Penalty Factor s_k		$5\times st_k$	$50\times st_k$	Mean	$5\times st_k$	$50\times st_k$	Mean	$5\times st_k$	$50\times st_k$	Mean
Prob. Size	Capacity									
Small: (N, K) = (10, 2)	C/0.6	4.67	15.36	10.02	3.50	7.72	5.61	0.86	4.56	2.71
	C/0.8	2.85	5.56	4.21	0.70	2.37	1.53	0.52	1.68	1.10
	C/1.0	1.35	1.86	1.61	0.00	0.63	0.32	0.21	0.32	0.26
	C/1.2	1.46	1.74	1.60	0.17	0.36	0.26	0.00	0.00	0.00
	Mean	2.58	6.13	**4.36**	1.09	2.77	**1.93**	0.40	1.74	**1.02**
Medium: (N, K) = (50, 10)	C/0.6	1.88	9.64	5.76	1.91	7.69	4.80	1.27	2.75	2.01
	C/0.8	1.25	5.21	3.23	1.11	4.50	2.80	0.62	2.41	1.51
	C/1.0	0.97	4.23	2.60	0.74	2.97	1.86	0.58	0.77	0.67
	C/1.2	1.13	2.73	1.93	0.89	1.95	1.42	0.88	2.58	1.73
	Mean	1.31	2.54	**3.38**	1.16	4.28	**2.72**	0.84	2.13	**1.48**
Large: (N, K) = (100, 20)	C/0.6	0.18	2.93	1.56	1.68	2.93	2.30	-0.01	1.41	0.70
	C/0.8	0.19	2.05	1.12	0.33	2.27	1.30	0.19	1.08	0.63
	C/1.0	-0.15	0.96	0.40	0.05	1.28	0.67	-0.15	0.96	0.40
	C/1.2	0.02	1.01	0.52	0.38	1.60	0.99	0.00	1.01	0.51
	Mean	0.06	1.74	**0.90**	0.61	2.02	**1.31**	0.01	1.11	**0.56**
Overall Mean (%)		1.32	4.44	**2.88**	0.95	3.02	**1.99**	0.41	1.63	**1.02**

Table 5: DH solution compared to the schedule practiced in a foundry (in item-days)

	DH Solution		Foundry Practice	
	Backlogs	Stocks	Backlogs	Stocks
Day 1	35,626	0	35,698	46
Day 2	25,564	0	25,324	23
Day 3	24,384	0	22,710	1
Day 4	6708	22	24,964	2
Day 5	3476	0	23,237	3
TOTAL	**95,758**	**22**	**131,933**	**75**

Table 6: DH solution compared to the schedule practiced in a foundry (in kg-days)

	DH Solution		Foundry Practice	
	Backlogs	Stocks	Backlogs	Stocks
Day 1	177,342	0	176,425	46
Day 2	115,244	0	120,117	23
Day 3	86,077	0	100,183	70
Day 4	67,009	47	101,314	140
Day 5	48,195	0	81,500	210
TOTAL	**493,867**	**47**	**579,540**	**489**