J Heuristics (2007) 13: 337–358
DOI 10.1007/s10732-007-9011-9

Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups

Silvio A. de Araujo · Marcos N. Arenales · Alistair R. Clark

Received: 1 December 2004 / Revised: 1 May 2006 /
Accepted: 9 May 2006 / Published online: 27 April 2007

Abstract A lot sizing and scheduling problem from a foundry is considered in which key materials are produced and then transformed into many products on a single machine. A mixed integer programming (MIP) model is developed, taking into account sequence-dependent setup costs and times, and then adapted for rolling horizon use. A relax-and-fix (RF) solution heuristic is proposed and computationally tested against a high-performance MIP solver. Three variants of local search are also developed to improve the RF method and tested. Finally the solutions are compared with those currently practiced at the foundry.

Keywords Lot-sizing · Scheduling · Sequence-dependent setup costs and times · Fix-and-relax · Local search

In many manufacturing systems, a key material is first prepared and then transformed into multiple products made from that material. Examples include the processing and mixing of liquids that constitute differently packaged drinks (Clark 2003b), and the production of metal alloys that are poured into different sand moulds in a foundry (Araujo et al. 2007; Santos-Meza et al. 2002). The scheduling of such systems must ensure the efficient coordination of the production of materials and their subsequent processing to make the products.

S.A. de Araujo
Universidade Estadual Paulista, Sao Jose do Rio Preto, Brazil

M.N. Arenales
Universidade de São Paulo, Sao Carlos, Brazil

A.R. Clark (✉)
Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol, BS16 1QY, UK
e-mail: Alistair.Clark@uwe.ac.uk

In practice, the lot sizing and scheduling of the products is carried out in a first phase taking into account the products' demand and due dates. The scheduling of material production is then determined in a second phase as a function of the outcome of the first phase. However, such an unintegrated approach can result in a poor schedule for the machines that prepare the materials.

This paper thus presents a model and solution methods for a manufacturing system where production planning has two linked levels: (1) the scheduling of machines with sequence-dependent setup costs and times to produce key materials which (2) are then used to manufacture final products. Section 1 reviews previous research. Section 2 proposes a mixed integer programming (MIP) model for integrated lot sizing and scheduling with backlogs and sequence-dependent material setup costs and times. The model is then adapted for rolling horizon use where only the immediate short-term schedules are implemented. Section 3 develops a relax-and-fix heuristic, solved using mathematical programming and three variants of local search: a descent heuristic, diminishing neighbourhood search and simulated annealing. The computational experiments are described in Sect. 4, including practical instances from a foundry. Conclusions are presented in Sect. 5.

1 Previous research

Numerous researchers have studied lot sizing and setup scheduling problems, with reviews by Drexl and Kimms (1997) and Karimi et al. (2003). The model put forward in this paper is closely related to the general lot sizing and scheduling problem (GLSP) and its setup time extensions, developed by Fleischmann and Meyr (1997), Haase and Kimms (2000) and Meyr (2000, 2002). The GLSP schedules multiple products on a single machine, permitting many setups in each single 'large-bucket' time period. This paper adapts the setup time version of GLSP to include backlogs and product-group setups, with an emphasis on rolling horizon use.

The review by Karimi et al. (2003) on capacitated lot sizing points out that there are not many papers about problems with multi-group joint setups, which is a feature of the model developed in this paper. Two such papers (Smith-Daniels and Smith-Daniels 1986; Smith-Daniels and Ritzman 1988) formulated models and methods for the case with sequence-dependent setup times, but were computationally viable only for small problems. More recently, Dumoulin and Vercellis (2000) proposed a MIP model for a three-level hierarchy of items, families, and types to represent both inter-family and inter-type setups. Unfortunately, the model is very difficult to solve even for very small instances, and so a viable Lagrangian decomposition and relaxation heuristic is developed. The Karimi et al. review highlights the development of heuristics with reasonable speed and solution quality for this kind of model as an important research area. Ghosh Dastidar and Nagi (2005) formulated a multi-workcenter model with family sequence-dependent setups, and developed an effective two-phase decomposition method that solved realistic problems to within a few percent of optimality in less than 20 minutes. The current paper's solution methods are a contribution in that same direction, but look to heuristically solve large problems by substituting most of the integer $0/1$ variables and constraints with continuous variables and constraints.

Several authors including Gopalakrishnan et al. (1995) and Porkka et al. (2003) have shown why it is important to not just explicitly include setup times in lot-sizing models, but to also properly model setup carryovers from one scheduling period to the next. Failure to do so can cause unnecessary setups, reduce effective capacity, and increase costs. The experimental results in Porkka et al. (2003) make a strong case for including setup carryover representation in MIP models (like the current paper) as opposed to the heuristic adjustment of solutions from models without such representation.

Problems with sequence-dependent setup times are particularly challenging. Arosio and Sianesi (1993) proposed a complex heuristic algorithm for simultaneous lot-sizing and sequencing on a single machine with sequence-dependent setups, but made no comparisons with optimal solutions in their computational tests. This paper does make such comparisons although, as will be seen, this is only possible for small problems. Ovacik and Uzsoy (1995) developed heuristics to minimise maximum lateness on parallel identical machines with sequence dependent setup times. They had some success with reaching a compromise between the impractical computational needs of optimal solutions and the poor quality of fast heuristic solutions by breaking a large problem into smaller ones which could be solved exactly. The basic method developed later in this paper also uses such a decomposition approach.

Belvaux and Wolsey (2001) presented and discussed formulations for a wide variety of practical lot sizing and scheduling problems with setups. They concluded that the large number of binary variables needed to effectively model changeovers (setups) continued to pose a challenge for solving the problems that arise in practice. This was also the experience of Haase and Kimms (2000) who were only able to solve problems varying from 3 items and 15 periods to 10 items and 3 periods. Wolsey (2002) identified large bucket models with sequence-dependent costs and/or times as still lacking tight mixed integer programming formulations that would permit optimal solutions for realistic sized problem instances using standard optimizers such as Cplex (ILOG 2001).

Recent research by Gupta and Magnusson (2005) into capacitated lot sizing with sequence-dependent costs (but with fixed setup times in order to avoid infeasibility) confirms that it is still difficult to obtain near-optimal solutions for industrial-size problems. A case in point is the work of Dillenberger et al. (1994) who 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. 2006), more widely known as *relax-and-fix* (Wolsey 1998; Kelly and Mann 2004). The current paper uses a similar basic approach as explained in Sect. 3. Dillenberger et al. (1994) 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.

This paper also develops a hybrid method that mixes *relax-and-fix* and local search on the combinatorial variables with linear programming on the continuous variables. The latter is used to calculate the objective value of the local search solution. Several authors have already explored this approach, among them Kuik et al. (1993) who used simulated annealing and tabu search on a lot-sizing problem with sequence-independent setups, and Teghem et al. (1995) who employed linear programming within a simulated annealing for a combinatorial production planning problem. Fleischmann and Meyr (1997) applied threshold accepting search to the GLSP with sequence-dependent setup costs (but not times). Meyr (2000) formulated the GLSPST (which considers both sequence-dependent setup costs and sequence-dependent setup times) and developed an efficient algorithm that uses multiple runs on a local search for lot sequencing with linear programming (LP) dual reoptimization for rapid lot-sizing. Meyr's model, however, did not allow backlogs of demand which is an important feature of the manufacturing systems considered in the current paper. Moreover, his algorithm was tested on problems with up to 18 products only. Nevertheless, the use of an LP within a local search is a powerful approach which this paper also follows and tests on problems larger than Meyr's. In Sect. 4, computational tests are also carried out comparing Meyr's GLSPST algorithm with the main approach developed in this paper.

Models such as Meyr (2000) and Dillenberger et al. (1994) are not only hard to solve optimally when the instances are large. They are also usually just temporary instances of the problem. Much of the data used in the model, for example forecasts of future demand, almost always changes as time rolls forward. The only decisions that are actually implemented are those in the time interval between the initial periods of successive optimizations. Production in later periods is only represented so that its impact on nearer and more immediate periods is taken into account.

The reality of such rolling horizon use of lot sequencing and sizing models motivated the following thinking: why specify detailed schedules for later periods if they are never implemented? Why not use a simplified representation for later periods in the rolling horizon that would be less difficult to solve and hence permit the solution of larger problems? Several authors have pursued this approach. Stadtler (2003) and Clark (2003a) 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 (2003) 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.

2 Problem definition and mathematical models

Consider a manufacturing system where materials are processed and then used to make final products. The system has the following characteristics and assumptions:

- A material may be used in multiple products, but a product is made from just one material.
- The output amount of a processed material is equal to the total amount of the products in which it is used.
- A product cannot be manufactured in a given time period unless the material from which it is made is also processed in that period. Thus processed materials cannot be held over to the next period. For example, in a foundry an alloy must be used while it is still liquid.

- In each time period only one material can be processed on a given finite-capacity machine (such as a furnace or mixer).
- A setup changeover from one material to another consumes capacity time in a manner that is dependent on the sequence in which the materials are processed. In addition, the triangle inequality holds, i.e., it is never faster to change over from one material to another by means of a third material. In other words, a direct changeover is at least as capacity efficient as going via another material.
- The setup state at the start of the first period is known and taken into account.
- 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.

To formulate a suitable mathematical model, consider the following indices and data:

Indices:	$j, k = 1, \dots, K$	processed materials.
	$p = 1, \dots, P$	products.
	$t = 1, \dots, T$	periods.
	$n = 1, \dots, \eta$	sub-periods.
Data:	C	Capacity available on the machine in each sub-period.
	ρ_p	Capacity required to produce one unit of product p.
	d_{pt}	Demand for product p in period t.
	$S(k)$	Set of products p that use material k. Each product uses just one material, i.e., $\{1, \dots, P\} = S(1) \cup \dots \cup S(K)$, and $S(k) \cap S(j) = \emptyset$, for all materials $k \neq j$, implying $\sum_k \lvert S(k) \rvert = P$.
	h^-_{pt}	Backlog penalty for delaying delivery of a unit of product p in period t.
	h^+_{pt}	Inventory penalty for holding a unit of product p in period t.
	s_{jk}	Setup penalty (or cost) for changing over from material j to material k, where $s_{jj} = 0$.
	st_{jk}	Setup time (loss of machine capacity) for changing over from material j to material k, where $st_{jj} = 0$.
Variables:	x_{pn}	Quantity (lot-size) of product p to be produced in sub-period n.
	I^+_{pt}	Inventory of product p at the end of period t, where I^+_{p0} is the starting inventory at the start of period 1.
	I^-_{pt}	Backlog of product p at the end of period t, where I^-_{p0} is the starting backlog at the start of period 1.
	y^k_n	Binary variable: $y^k_n = 1$ if the machine is configured for production of material k in sub-period n, otherwise $y^k_n = 0$.
	z^{jk}_n	Binary setup variable: $z^{jk}_n = 1$ if there is a machine changeover from material j to material k at the start of sub-period n, otherwise $z^{jk}_n = 0$. Thus $z^{jk}_n = 1$ if $y^j_{n-1} = 1$

& $y_n^k = 1$, and $z_n^{jk} = 0$ if $y_{n-1}^j = 0$ or $y_n^k = 0$. It is relaxed to be continuous for reasons explained below.

In addition, the following definitions from the General Lot-Sizing and Scheduling Problem (GLSP) model are used:

η_t	Number of sub-periods in period t.
$F_t = 1 + \sum_{\tau=1}^{t-1} \eta_\tau$	The index of the first sub-period in period t. Note that $F_1 = 1$.
$L_t = F_t + \eta_t - 1$	The index of the last sub-period in period t. Note that $L_1 = \eta_1$.
$\eta = \sum_{t=1}^{T} \eta_t$	Total number of sub-periods over periods $1, \dots, T$.

Typically, a period t corresponds to a workday and in this case $\eta_t = L$, the number of loads that the machine can process in a workday.

2.1 Overall model

Thus a mathematical model for scheduling product lot sizes and setups over a planning horizon of T periods can be formulated as:

$$\text{Minimize} \quad \sum_p \sum_t (h_{pt}^- I_{pt}^- + h_{pt}^+ I_{pt}^+) + \sum_j \sum_k \sum_{n=F_1}^{L_T} s_{jk} z_n^{jk} \tag{1}$$

$$\text{subject to} \quad I_{p,t-1}^+ - I_{p,t-1}^- + \sum_{n=F_t}^{L_t} x_{pn} - I_{pt}^+ + I_{pt}^- = d_{pt}, \quad \forall p, t, \tag{2}$$

$$\sum_{p \in S(k)} \rho_p x_{pn} + st_{jk} z_n^{jk} \le C y_n^k, \quad \forall j, k, n = F_1, \dots, L_T, \tag{3}$$

$$z_n^{jk} \ge y_{n-1}^j + y_n^k - 1, \quad \forall j, k, n = F_1, \dots, L_T, \tag{4}$$

$$\sum_k y_n^k = 1, \quad \forall n = F_1, \dots, L_T, \tag{5}$$

$$y_n^k \in \{0, 1\} \quad \text{with } y_0^k = 0, \quad \forall k, n = F_1, \dots, L_T, \tag{6}$$

$$z_n^{jk} \ge 0, \quad \forall j, k, n = F_1, \dots, L_T, \tag{7}$$

$$x_{pn} \ge 0 \quad \text{and integer}, \quad \forall p, n = F_1, \dots, L_T, \tag{8}$$

$$I_{pt}^+ \quad \text{and} \quad I_{pt}^- \ge 0, \quad \forall p, t. \tag{9}$$

The objective function (1) minimizes a weighted sum of inventory & backlog penalties over each period and sequence-dependent setup penalties. Constraints (2) balance inventories, backlogs, demand and production in each period. Constraints (3) not only keep production within the machine's capacity, but also ensure that only products of the same material are produced in a particular machine loading.

As y_n^k and y_{n-1}^j are both binary variables, constraints (4) and the objective function (1) force the continuous variable z_n^{jk} to have value 1 if there is a changeover from material j to material k and, along with constraints (7), to have value 0 otherwise.

Constraints (5) and (6) ensure that there is only a single machine loading in each sub-period.

The model assumes that the system is not initially setup for any material, and that every setup must be completed in a sub-period. This means that the capacity time st_{jk} needed to setup a material k must not be greater than C, the capacity available on the machine in each sub-period.

Lot-sizing models generally consider the lot-size variables x_{pn} to be continuous. If x_{pn} is large, then simply rounding to the nearby integer value is usually acceptable. However, for problems where capacity or demand is low then the integer conditions (8) on x_{pn} must hold. Constraints (9) measure inventory I^+_{pt} and backlogs I^-_{pt} as non-negative variables, but for a given pair (p,t), I^+_{pt} and I^-_{pt} will not both be strictly positive in a optimal solution, given their positive coefficients in the objective function (1).

The model draws upon ones with sequence-dependent setups previously proposed by Smith-Daniels and Smith-Daniels (1986) and Meyr (2000), but with the difference that backlogs as well as finished inventories are represented. As with the GLSP, sequencing decisions in the model are implicitly decided by the machine configuration variables y^k_n. However, in common with Smith-Daniels and Smith-Daniels (1986), but in contrast to the GLSP where a setup relates to only one product, a setup in our model is associated with a specific material, permitting the manufacture of any product that uses that material. Hence a sub-period in our model is similar, but different, to the GLSP concept of a variable number of *small buckets* of time of flexible length (Drexl and Kimms 1997). It is different because:

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

The result is a joint scheduling model for material processing (y^k_n, z^{jk}_n) and lot sizing (x_{pn}).

Lot-sizing MIP models can be a challenge even for industrial-strength solvers such as Cplex (ILOG 2001), which are unable to optimally solve many real problems in a practical amount of time. An attempt was made, as an initial benchmark method, to solve model (1–9) with realistic foundry data using Cplex 7.1 on a Pentium III 500 MHz. The default branch-and-cut search soon exhausted the computer's 512 MB of RAM without converging to optimality, terminating with a sizeable gap between the best solution found (the incumbent) and the highest lower bound.

As discussed at the end of Sect. 1, the model's input data is generally imprecisely known and unstable in practice, so it is not worthwhile to spend a lot of time seeking an optimal solution. To do so would be to exactly solve the wrong problem. Backlog penalties tend to be poorly defined subjective estimates and the order book is usually updated daily, so an optimal solution based on forecast demand and estimates of penalties will almost certainly be an approximate solution for the real problem. A more useful result will be a rapidly found solution of good quality. For that reason, a rolling horizon method (Araujo et al. 2007; Clark 2005) is applied, implementing the immediate decisions relating to the next one or few periods, after which the planning horizon is rolled forward and the model is applied once more with updated order and inventory information.

2.2 Rolling horizon model

Suppose each period t is a workday, as in the foundry that motivated this paper. 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 $L = \eta_1 = 10$ sub-periods, as up to L material loadings can be processed each day. The remaining days $t = 2, \ldots, 5$ have just one sub-period each ($\eta_2 = \eta_3 = \eta_4 = \eta_5 = 1$). Thus $F_1 = 1$; $L_1 = 10$; $F_2 = 11 = L_2$; $F_3 = 12 = L_3$; $F_4 = 13 = L_4$; $F_5 = 14 = L_5$, i.e., there are $\eta = 14$ sub-periods n (as illustrated in Fig. 1). The variables y_n^k for the larger sub-periods $n = F_2, \ldots, F_5$ are then redefined as "the number of loadings using material k produced in sub-period n".

Only the scheduled decisions relative to the $\eta_1 = 10$ 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 (Clark 2005).

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

The outcome is the following model for rolling horizon use, denominated RH:

Model RH:

$$\text{Minimize} \quad \sum_p \sum_t (h_{pt}^- I_{pt}^- + h_{pt}^+ I_{pt}^+) + \sum_j \sum_k \sum_{n=F_1}^{L_1} s_{jk} z_n^{jk} \tag{10}$$

$$\text{subject to} \quad I_{p,t-1}^+ - I_{p,t-1}^- + \sum_{n=F_t}^{L_t} x_{pn} - I_{pt}^+ + I_{pt}^- = d_{pt}, \quad \forall p, t, \tag{11}$$

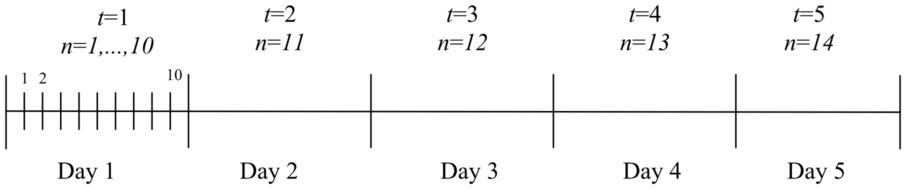

Fig. 1 Periods and sub-periods in a rolling horizon strategy

$$\sum_{p\in S(k)} \rho_p x_{pn} + st_{jk} z_n^{jk} \le C y_n^k, \quad \forall j,k,n = F_1,\dots,L_1, \tag{12}$$

$$\sum_{p\in S(k)} \rho_p x_{pn} \le C y_n^k, \quad \forall k,n = F_2,\dots,L_T, \tag{13}$$

$$\sum_k y_n^k = \frac{L}{\eta_t}, \quad \forall t,n = F_t,\dots,L_t, \tag{14}$$

$$z_n^{jk} \ge y_{n-1}^j + y_n^k - 1, \quad \forall j,k,n = F_1,\dots,L_1, \tag{15}$$

$$y_n^k \in \{0,1\} \quad \text{with } y_0^k = 0, \quad \forall k,n = F_1,\dots,L_1, \tag{16}$$

$$y_n^k \ge 0 \quad \text{and integer}, \quad \forall k,n = F_2,\dots,L_T, \tag{17}$$

$$z_n^{jk} \ge 0, \quad \forall j,k,n = F_1,\dots,L_1, \tag{18}$$

$$x_{pn} \ge 0 \quad \text{and integer}, \quad \forall p,n = F_1,\dots,L_1, \tag{19}$$

$$x_{pn} \ge 0, \quad \forall p,n = F_2,\dots,L_T, \tag{20}$$

$$I_{pt}^+ \quad \text{and} \quad I_{pt}^- \ge 0, \quad \forall p,t. \tag{21}$$

Constraints (3) are now replaced by (12) for the first period and (13) for the remaining periods. Constraint (5) is replaced by (14) which imposes exactly L/η_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 $L/\eta_1 = 10/10 = 1$, whereas period 2 has one sub-period and can handle 10 loads, so $L/\eta_2 = 10/1 = 10$.

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.

A solution to model RH is not a solution to model (1–9), as just the first loadings are scheduled and actually implemented, while the other days are planned only approximately. However, the application T times of model RH, starting consecutively at periods $1, 2, \dots, 5$, 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 (2003). The final objective function value is gradually accumulated over the T applications of model RH, each of which contributes its period 1 part of expression (10), i.e., excluding the part for periods 2 onwards: $\sum_p \sum_{t=2}^T (h_{pt}^- I_{pt}^- + h_{pt}^+ I_{pt}^+)$.

3 Solution methods

Although model RH is much smaller than model (1–9), it is still not small enough to be solved optimally with realistic data within acceptable computing time. However, it is possible to solve model RH in two steps using the *relax-and-fix* method (Wolsey 1998), as follows:

Step 1. *Relax* all integer variables, except the first day's binary variables y_n^k for $n = F_1, \ldots, L_1$, these being the most important decisions in the rolling horizon method. Solve this relaxed problem.

Step 2. *Fix* the first day's binary variables y_n^k at their solution values in step 1. Respecify as integer the y_n^k variables for the subsequent days ($n = F_2, \ldots, L_T$) and as integer the x_{pn} variables for the first day ($n = F_1, \ldots, L_1$). Solve this partially fixed problem.

The problem in step 1 is still difficult to solve exactly, and so is solved using one of the four heuristics methods (RF, DH, DN and SA) described in Sects. 3.1 to 3.4 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 (14), that $x_{pn} = 0$ for all $p \notin S(k)$, 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.

3.1 Basic relax-and-fix (RF) method

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 5, 10 and 20 minutes respectively for small, medium and large problems (as defined in Table 1 of Sect. 4.1). This method is denoted RF.

3.2 Descent heuristic (DH)

Step 1 of the *relax-and-fix* method can also be solved using a local search *descent heuristic* (Aarts and Lenstra 1997) to find the values of the first day's binary variables y_n^k for $n = F_1, \ldots, L_1$. The search starts with an initial current solution (obtained, for example, by random selection) whose objective function value is obtained by solving the resulting linear programme (LP). The current solution values of the y_n^k binary variables for day 1 are then randomly modified (as explained below) to provide a neighbouring solution whose objective function value is calculated by using the dual simplex method to efficiently resolve the modified LP. If it is an improvement, the neighbouring solution is adopted to be the current one. Experiments showed that 1000 iterations are sufficient to obtain a good solution within an acceptable running time.

The solution representation in the descent heuristic consists of a η_1-vector of integers, $\boldsymbol{v} = (v_1, \ldots, v_{\eta_1})$, 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 $\eta_1 = 10$, 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. Observe that only one material k is changed in a single sub-period n.

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

1. For $n = 1, \ldots, \eta_1$, choose the value of v_n to be k with the probability given by $|S(k)|/P$, i.e., the more products that can be made from k, the more likely k will be selected.

2. Run a MIP solver for a few minutes to obtain an initial heuristic solution.
3. For $n = 1, \dots, \eta_1$, choose the value of v_n to be k, uniformly sampled from $\{1, \dots, K\}$.

However, after initial tests the first way was selected and the other two were discarded before the computational testing of Sect. 4.

Two different neighbourhood moves were implemented, as follows.

MOVE 1: The first move slightly biases the selection towards more widely-used materials, and was adopted after initial testing showed its positive impact. Two alternative procedures were used to randomly choose n^*, the sub-period in which to change the material k:

1. With 90% probability: The value of n^* is uniformly sampled from the set $\{1, \dots, \eta_1\}$.
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 $(P - |S(k)|)/(P(K-1))$. If material k is not produced in period 1, then another value of k must be sampled. After this, the value of n^* is 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, randomly chosen with probabilities 90% and 10% respectively:

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

MOVE 2: The second move tries to minimize the setup costs (the third part of the objective function (10)) and setup time. Two randomly selected positions in the array are swapped, thus changing the sequence of material production.

At each search iteration i, MOVE 2 is selected with probability $(i/IT)^2$ where $IT = 1000$ is the total number of iterations. Thus for the first 70.7% iterations, MOVE 1 is used with higher probability than MOVE 2 in order to look for widely-used materials to produce. As the search progresses the probability of MOVE 2 is increased, so that the final neighbourhood moves are mainly seeking to resequence production. The rationale for this is that after many iterations the "best" material types will generally have been selected and so focus is switched to swapping positions in order to minimize sequence-dependent material changes.

Many researchers (Diaz et al. 1996; Aarts and Lenstra 1997; Glover and Kochenberger 2003) have proposed various ways to improve descent heuristic performance through avoidance of entrapment at a local optimum (i.e., where no better solution exists within the defined neighbourhood). In this paper, two strategies are used: Diminishing Neighbourhood Search and Simulated Annealing, with the same basic parameters as the descent heuristic described above.

3.3 Diminishing neighbourhood (DN) search

This method adapts the local search descent heuristic described above, 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 a wasteful search. The search starts with the largest possible neighbourhood, i.e., all the first day's η_1 variables y_n^k in a solution can be changed. After a certain number of iterations the neighbourhood size is reduced, i.e., only $\eta_1 - 1$ randomly uniformly selected variables y_n^k in a solution can change. 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 Sect. 3.2.

For each size $N = 1, \dots, 10$ of neighbourhood, $18(11 - N) + 1$ iterations are carried out, summing to a total of 1000 iterations over the whole search. Thus 19 iterations are carried out when $N = 10$ at the start of the search, 37 when $N = 9$, and so on, increasing to 181 iterations when $N = 1$ at the end of the search. Clark (2003b) successfully used a similar method for lot-sizing and scheduling on a drinks canning line.

3.4 Simulated annealing (SA)

Simulated Annealing (Eglese 1990) 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:

$$p(\Delta\text{ofv}) = e^{-(\frac{\Delta\text{ofv}}{\text{Temp}})}, \tag{22}$$

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

Initial computational tests indicated that the best results were obtained with the following parameters. The starting temperature $\text{Temp}_{\text{start}}$ is a function of the initial solution (Diaz et al. 1996):

$$\text{Temp}_{\text{start}} = \frac{\mu \times \text{Value of the Initial Solution}}{-\log(\theta)}, \tag{23}$$

where $\mu = 0.6$ and $\theta = 0.9$ 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:

$$\text{Temp}_{\text{new}} = \text{Temp}_{\text{old}} - \left(0.1 \times \text{Temp}_{\text{old}} \times \frac{\Delta\text{ofv}}{\text{ofv}(S)}\right), \tag{24}$$

where ofv(S) was the objective function value of 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.

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

Parameters	Values
Number of Products and Materials: (P, K) pairs	Small Problem: (10, 2)
	Medium Problem: (50, 10)
	Large Problem: (100, 20)
Number of Periods (T)	5
Inventory Penalty (h^+_{pt})	[2, 10]
Backlog Penalty (h^-_{pt})	[20, 100]
Necessary capacity to produce one unit of product p (ρ_p)	[0.1, 3]
Setup Time (st_{jk})	[5, 10]
Setup Penalty (s_{jk})	Low: $50 \times st_{jk}$
	High: $500 \times st_{jk}$
Demand (d_{pt})	[40, 60]
Tightness of Capacity (C)	Loose: LFLC/0.6
	Tight: LFLC/0.8
	Too Tight: LFLC/1.0

4 Computational results

This section first describes how the test data was generated. It goes on to analyse the quality of the RF method, and then evaluates the three local search methods. It concludes by comparing the methods with the scheduling practiced in a foundry.

4.1 Data generation

The tests were carried out with $T = 5$ periods divided into 10 sub-periods each. The parameter values used to generate random test examples, shown in Table 1, are based on Haase and Kimms (2000).

Previous experience (Clark 2005) indicates that certain parameters may affect solution quality, namely:

- Problem size (P, K)
- Size of setup penalty s_{jk}
- Tightness of capacity C

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 tightness of capacity C was based on a mean value calculated from a lot-for-lot (LFL) policy, ignoring setup times. The lot-for-lot capacity, denoted by LFLC, required to produce exactly the total demand in each period was averaged over all periods and machine loadings, to give:

$$\text{LFLC} = \frac{\sum_{p=1}^{P}\sum_{t=1}^{T} d_{pt}\rho_p}{T \times L}. \tag{25}$$

Thus in Table 1 three different levels of the tightness of capacity C are shown: (i) Loose: $C = \text{LFLC}/0.6$; (ii) Tight: $C = \text{LFLC}/0.8$; (iii) Too Tight: $C = \text{LFLC}/1.0$.

The parameters (P, K) pairs, s_{jk} and C were varied in a 3-factor experimental design. All the factorial combinations were randomly replicated 10 times, resulting in a total of $3 \times 2 \times 3 \times 10 = 180$ test problems, each of which was solved to provide a complete 5-day schedule for model (1–9), using the following methods:

(a) The Cplex solver's direct attempt at an optimal solution, after running its general-purpose branch-and-cut algorithm for a given amount of time.
(b) The basic relax-and-fix method (RF).
(c) The three local search methods (DH, DN and SA).

The solution values presented below for methods (b) and (c) are calculated as described at the end of Sect. 2.2, i.e., the accumulation of the period 1 part of expression (10) over the T applications of model RH.

4.2 Evaluation of the RF method

The RF method was assessed using the solutions and lower bounds provided by the Cplex method (a) above. This was motivated by the favorable test results obtained in Araujo et al. (2007) where setup times were not sequence-dependent, in contrast to this paper. In Araujo et al. (2007), tests on 240 problems (with varying characteristics of size, capacity tightness and setup time) showed that on average, the RF method solution was only 8.6% from the Cplex-supplied lower bounds, while the Cplex incumbent itself was 22.3% from these same bounds. A similar range of tests was carried out for this paper's problems with sequence-dependent setup times. To ensure a fair comparison between the RF and Cplex solutions, the setup state was preserved from the end of one period to the beginning of the next when implementing the RF method, i.e., the initial setup state in a given period was the same as the final setup state of the previous period's optimization, different from the null setup state $y_0^k = 0$ specified in constraints (16).

For small problems with loose or tight capacity, Cplex always found a near-optimal solution within 5 minutes. However for all medium and large-sized problems, the time limit of 1 hour was always reached before proving optimality, obliging the incumbent to be used as a heuristic solution.

Table 2 shows the mean solution disparity of the Cplex and RF methods compared to the lower bounds provided by the Cplex method branch-and-cut search. The disparity is calculated as:

$$\text{Disparity} = \frac{\text{Method Solution} - \text{Lower Bound}}{\text{Method Solution}} \times 100\%.$$

Cplex produces better quality solutions than the RF method for small problems when capacity is not too tight. However, for medium and large problems, the comparison is clouded by the prevalence of 99% disparity values. These are caused by relatively small lower bounds and indicate the struggle Cplex has in proving optimality. Nevertheless, Table 2 shows that the RF method generally supplies better solutions than Cplex for medium and large problems. The raw solution values and RF % disparities in Table 3 demonstrate this more clearly, particularly for large problems.

Table 2 Solution % disparity from the Cplex Lower Bound

Method		Cplex		RF	
Problem Size	Capacity	Small s_{jk}	Large s_{jk}	Small s_{jk}	Large s_{jk}
	Loose	0.32	1.23	0.32	34.24
Small	Tight	1.94	0.54	1.94	26.90
	Too Tight	26.84	4.46	4.07	4.55
	Loose	99.86	99.85	99.85	99.85
Medium	Tight	97.60	99.21	97.04	99.02
	Too Tight	73.14	87.30	68.68	83.35
	Loose	99.85	96.41	97.31	96.60
Large	Tight	99.86	99.87	98.04	98.59
	Too Tight	99.49	99.50	95.91	96.51

Table 3 Cplex & RF Solution Values with Cplex Lower Bounds & RF % disparity

		Cplex		RF		Cplex Lower Bound		(RF-Cplex) / Cplex	
Problem Size	Capacity	Small s_{jk}	Large s_{jk}	Small s_{jk}	Large s_{jk}	Small s_{jk}	Large s_{jk}	Small s_{jk}	Large s_{jk}
	Loose	1.961	13.054	1.961	19.606	1.954	12.894	0%	50%
Small	Tight	1.961	14.416	1.961	19.615	1.922	14.338	0%	36%
	Too Tight	7.539	23.347	5.749	23.370	5.515	22.306	−24%	0%
	Loose	24.171	139.805	22.082	133.950	33	207	−9%	−4%
Medium	Tight	43.638	170.848	35.373	137.603	1.045	1.352	−19%	−19%
	Too Tight	120.204	253.855	103.074	193.694	32.285	32.250	−14%	−24%
	Loose	3,646.046	319.793	208.872	338.364	5.628	11.494	−94%	6%
Large	Tight	4,536.679	4,693.047	312.100	435.437	6.132	6.132	−93%	−91%
	Too Tight	4,536.679	4,693.047	568.719	667.753	23.285	23.285	−87%	−86%

The RF % disparity is the percentage by which the RF method lowers the objective value compared to the Cplex solution:

$$\text{RF \% Disparity} = \frac{\text{RF Solution} - \text{Cplex Solution}}{\text{Cplex Solution}} \times 100\%$$

and brings out the great advantage of the RF method over Cplex for large problems with tight capacity, the most challenging for industry.

Lower bounds for model (1–9) can be improved by considering the relaxation where all the setup penalties s_{jk} and times st_{jk} are set at their respective minimum values $\min_{(j,k)}\{s_{jk}\}$ and $\min_{(j,k)}\{st_{jk}\}$. Although not trivial, the relaxed problem is now much easier to solve (as the setups are sequence-independent) and its optimal objective function value provides an alternative lower bound. Judging by similar tests in Araujo et al. (2007) the Cplex incumbent for this problem is likely to be of good quality. Although there is no guarantee that this "minimum setup" incumbent is optimal and hence a lower bound for model (1–9), it provides a better indication of quality of the Cplex and RF solutions than the Cplex lower bounds applied to the problem where setups are sequence-dependent. Over the 180 test problems, the Cplex and

RF solutions were on average 13 and 46% respectively above the "minimum setup" incumbent". These proxy "lower bounds" are much tighter than the Cplex ones in Table 2, and provide a far clearer indication of the quality of the solutions.

To try to evaluate the RF method against a good heuristic from the literature, the availability of executable code for the GLSPST algorithm (Meyr 2000) enabled a partial comparison to be made. However, the following differences must be borne in mind:

- Meyr does not consider product families, so the comparison had to specify setup costs $s_{pq} = 0$ and setup times $st_{pq} = 0$ for all products p and q in the same family $S(k)$, i.e., using the same material k. Furthermore $s_{pq} = s_{jk}$ and $st_{pq} = st_{jk}$ for all $p \in S(j)$, $q \in S(k)$, where $k \neq j$.
- In Meyr's model, stock holding costs do not vary over time, i.e., $h^{+}_{pt} = h_p$ for all products p.
- Meyr's model admits no backlogs, i.e., $I^{-}_{pt} = 0$ for all products p and macro-period t.
- The capacity K_t in Meyr is defined over each macro-period t, rather than over a micro-period.
- Meyr's algorithm was run with its supplied default parameters.

Moreover and most importantly, Meyr's model is more flexible than (1–9) for the following reasons.

- The lot size is model (1–9) is integer whereas it is continuous in Meyr's model.
- In model (1–9) the duration of a sub-period is constant whereas in Meyr (2000) it is variable, being determined by the lot-size.

The results in Table 4 show the disparity of the RF solutions compared to Meyr's:

$$\text{Disparity} = \frac{\text{RF Solution} - \text{Meyr Solution}}{\text{RF Solution}} \times 100\%.$$

Meyr's algorithm found a feasible solution for all instances with Loose and Tight capacity, but never when capacity was Too Tight (as no feasible solutions exist when backlogs are not allowed). For small instances, method RF performed better than

Table 4 Solution % disparity of the RF solution from the Meyr Heuristic solution

Problem Size	Capacity	Small s_{jk}	Large s_{jk}
	Loose	−3.74	28.76
Small	Tight	−1.65	18.40
	Too Tight	Meyr is infeasible	Meyr is infeasible
	Loose	−20.57	−29.29
Medium	Tight	20.60	−46.14
	Too Tight	Meyr is infeasible	Meyr is infeasible
	Loose	5.03	−0.15
Large	Tight	32.07	−34.97
	Too Tight	Meyr is infeasible	Meyr is infeasible

Table 5 Mean solution disparity (%) of the DH/DN/SA heuristics compared to the basic RF method

Method		DH		DN		SA	
Prob. Size	Capacity	Low s_{jk}	High s_{jk}	Low s_{jk}	High s_{jk}	Low s_{jk}	High s_{jk}
	Loose	18.32	4.08	0.00	0.03	0.00	0.02
Small	Tight	79.36	10.66	6.15	−0.01	0.00	0.00
	Too Tight	105.0	25.39	1.46	−0.94	0.38	2.25
	Loose	18.23	11.22	−1.39	−5.42	0.70	−5.12
Medium	Tight	11.36	15.86	1.74	−2.01	0.76	−0.85
	Too Tight	0.66	8.83	−8.04	0.03	−2.93	−1.37
	Loose	2.17	6.63	1.30	0.42	−1.68	0.62
Large	Tight	0.99	5.15	−1.12	−0.38	−2.80	−0.56
	Too Tight	0.58	0.10	−3.55	−6.79	−4.66	−3.37

Meyr's algorithm for small setup penalties, but not for large penalties. For medium instances, method RF performed better except for small setups under tight capacity. For large instances, method RF performed better for large setups but not for small ones. Although not clear-cut, these results indicate that the RF method is reasonably competitive, despite the greater flexibility of Meyr's model and faster speed of his algorithm.

4.3 Evaluation of the DH, DN and SA methods

Table 5 shows the mean solution disparity of the DH, DN and SA local search methods relative to the basic RF method, calculated as:

$$\text{Disparity} = \frac{(\text{Heurístic Solution} - \text{RF Solution})}{\text{RF Solution}} \times 100\%. \tag{26}$$

For small problems, the DH method produces poor solutions that can be over 100% worse than RF (which, for small problems, finds the optimal solution in step 1 of the *relax-and-fix* algorithm). The DN and SA methods find the optimal solution for nearly all the small problems. Negative values were obtained for these methods due to the fact that Cplex was set to consider as optimal any solution within 1% of its lower bound.

For medium and large problems, the disparity of the DH, DN and SA methods is reduced. Recall that for problems of these sizes, Cplex was unable to prove optimality in step 1 of the *relax-and-fix* algorithm, thus weakening the basic RF method. This means that the DH, DN and SA methods are not being compared against provably-optimal MIP solutions, but against possibly suboptimal incumbent solutions obtainable within the preset Cplex time limits.

The difference in relative performance of the DH, DN and SA methods might be explained as follows. For problems of all sizes, both DH and SA will initially follow the same search path, but when either reaches a local optimum, DH has no way to jump out of it and so the 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

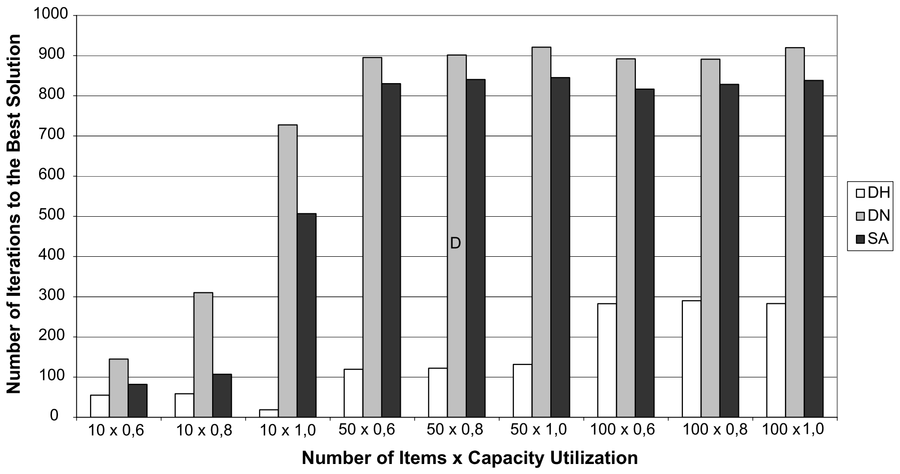

Fig. 2 Mean number of iterations before each method reaches its best solution

path than DH and SA, and if it gets stuck at a local optimum, this tends to be when its neighbourhood is near its minimum size towards the end of the search.

For small problems, the tighter capacity utilization, the larger the disparity of the DH method, while there is little effect on the DN and SA methods. However, for medium and large problems, an increase in capacity tightness results in smaller disparity for DH, DN and SA methods. This might be partly explained by the same reason noted above, that is, the larger the size of the problem and the tighter capacity, the more difficult the problem is to solve, and the worse the performance of Cplex.

The DN and SA modifications of the descent heuristic help to prevent the search getting stuck at local optima. Figure 2 shows the mean number of iterations before each method reaches its best solution. Note that the number of iterations for the DN and SA methods tends to be much larger than for the DH method, showing that the DN and SA methods improve the solution throughout the search, avoiding bad local optima, while the DH method soon stops at a local optimum. Figure 3 shows the mean times before each method reaches its best solution, confirming that larger problems do indeed need more computing time with similar number of iterations as smaller problems. The less than 10 minutes needed by the DN and SA methods for larger problems is practicable for industrial use.

Figure 4 shows the progress of the searches, i.e., the solution value over time of the DH, DN and SA methods. Observe that the DH method rapidly converges to a local optimum and then stagnates while the other methods continue for many more iterations before converging. The DN method also tended to quickly identify a good schedule.

The DH, DN and SA methods took about 2, 4 and 10 minutes to solve step 1 of the relax-and-fix for small, medium and large problems respectively. These are viable times and faster than the respective 5, 10 and 20 minutes time limits of the basic RF method of Sect. 3.1. For small problems, the basic RF method very quickly found the optimal solution, but as the number of products increased, it soon reached its time

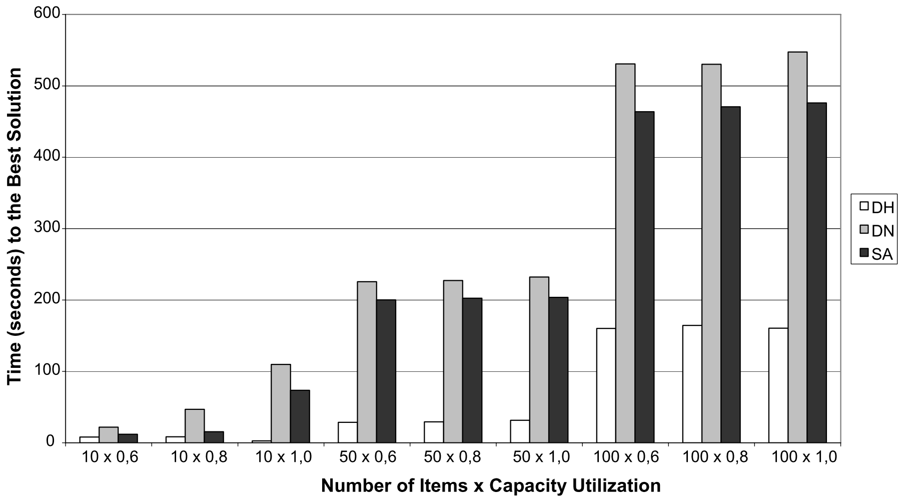

Fig. 3 Mean time (in seconds) before each method reaches its best solution

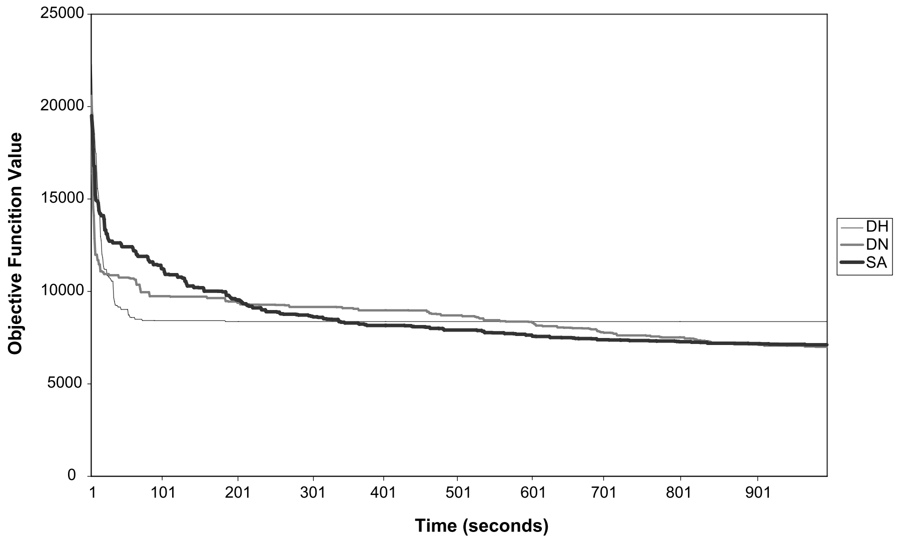

Fig. 4 Behaviour of each method over the search time. (Example of a medium-sized problem with loose capacity and large setup times)

limit before finding a provably optimal solution. Neither tightness of capacity nor setup penalty influenced computing time.

The solution of the final MIP (step 2) 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, almost certainly because the binary variables y_n^k $(n = F_1, \dots, L_1)$ had been fixed, leaving only the non-zero integer x_{pn} values to be optimized.

4.4 Evaluation in the foundry

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

The solutions output by the DH, DN and SA method were almost identical and 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 1000 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 multiplying the quantity (weight) by the number of days in stock or backlogged. Tables 6 and 7 show the 5 day schedules output by the DH method compared to those scheduled manually. The "Day 5" line at the end of the DH method's schedule shows that the final stocks and backlogs were respectively 0 and 3,476 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 substantially reduced backlogs and delays.

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 as opposed to 28].

Table 6 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	6,708	22	24,964	2
Day 5	3,476	0	23,237	3
Total	95,758	22	131,933	75

Table 7 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

Note from Tables 6 and 7 that the 5-day totals of the DH backlogs and stocks are substantially lower, i.e., respectively 27 & 90% by weight, and 15 & 71% by number of items. This reflects the foundry's concern to maximize utilization of capacity, a policy which tended to prioritize larger lot-sizes, resulting in the repeated postponement or early production of orders, thus creating the larger backlogs and inventories observed in Tables 6 and 7.

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

5 Conclusion

This paper developed a mixed integer linear programme (MIP) model for lot sizing and scheduling that considers backorders and sequence-dependent setup costs and times for group changeovers. With the exception of small problems, it was not possible to optimally solve the model within reasonable computing time, even using an advanced MIP solver such as Cplex 7.1. In order to efficiently identify an approximate solution, a rolling horizon model and associated *relax-and-fix* (RF) procedure were developed. Computational tests showed the basic RF method produced much better solutions on medium and large sized problems than Cplex, particularly for realistic situations of tight capacity. A comparison with Meyr (2000)'s algorithm showed the basic RF method to be reasonably competitive, particularly on medium and large problems under tight capacity and with large setup penalties.

In order to better solve step 1 of the *relax-and-fix* method, three variants of local search were developed: Descent Heuristic (DH), Diminishing Neighbourhood (DN) and Simulated Annealing (SA). Computational tests showed the DN and SA methods gave better results than the basic RF method, particularly under tight capacity on medium and large problems. The DH, DN and SA methods all produced much improved schedules in much less time compared to the schedules created manually in the foundry that motivated this paper.

Overall, this paper has demonstrated that a hybrid of mathematical programming and local search methods are able to generate good quality lot sizing and sequencing schedules for challenging industrial-sized problems encountered in practice.

Acknowledgements The authors would like to thank the referees for their very helpful feedback, resulting in a better paper, and are also grateful to Herbert Meyr for making available the executable code of his heuristic.

References

Aarts, E.H.L., Lenstra, J.K.: Local Search in Combinatorial Optimization. Wiley, Chichester (1997)

Araujo, S.A., Arenales, M.N., Clark, A.R.: A Lot-sizing and furnace scheduling in small market-driven foundries. Comput. Oper. Res. (2007, in press)

Arosio, M., Sianesi, A.: A heuristic algorithm for master production scheduling generation with finite capacity and sequence dependent setups. Int. J. Prod. Res. **31**, 531–553 (1993)

Belvaux, G., Wolsey, L.A.: Modelling practical lot-sizing problems as mixed integer programs. Manag. Sci. **47**, 993–1007 (2001)

Beraldi, P., Ghianib, G., Guerrieroc, E., Grieco, A.: Scenario-based planning for lot-sizing and scheduling with uncertain processing times. Int. J. Prod. Econ. **101**(1), 140–149 (2006)

Clark, A.R.: Optimization approximations for capacity constrained material requirements planning problems. Int. J. Prod. Econ. **84**, 115–131 (2003a)

Clark, A.R.: Hybrid heuristics for planning lot sizes and setups. Comput. Ind. Eng. **45**, 545–562 (2003b)

Clark, A.R.: Rolling horizon heuristics for production and setup planning with backlogs and error-prone demand forecasts. Prod. Plan. Control **16**, 81–97 (2005)

Diaz, A., Glover, F., Ghaziri, H.M., González, J.L., Laguna, M., Moscato, P., Tseng, F.T., Optimización Heurística y Redes Neuronales. Editorial Paraninfo, Spain (1996)

Dillenberger, C., Escudero, L.F., Wollensak, A., Zhang, W.: On practical resource allocation for production planning and scheduling with period overlapping setups. Eur. J. Oper. Res. **75**, 275–286 (1994)

Drexl, A., Kimms, A.: Lot sizing and scheduling—survey and extensions. Eur. J. Oper. Res. **99**, 221–235 (1997)

Dumoulin, A., Vercellis, C.: Tactical models for hierarchical capacitated lot-sizing problems with setups and changeovers. Int. J. Prod. Res. **38**, 51–67 (2000)

Eglese, R.W.: Simulated annealing: a tool for operational research. Eur. J. Oper. Res. **46**, 271–281 (1990)

Fleischmann, B., Meyr, H.: The general lotsizing and scheduling problem. Oper. Res. Spectr. **19**, 11–21 (1997)

Ghosh Dastidar, S., Nagi, R.: Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. Comput. Oper. Res. **32**, 2987–3005 (2005)

Glover, F.W., Kochenberger, G.A. (eds.): Handbook of Metaheuristics. Kluwer Academic, Boston (2003)

Gopalakrishnan, M., Miller, D.M., Schmidt, C.P.: A framework for modelling setup carryover in the capacitated lot-sizing problem. Int. J. Prod. Res. **33**, 1973–1988 (1995)

Gupta, D., Magnusson, T.: The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. Comput. Oper. Res. **32**, 727–747 (2005)

Haase, K., Kimms, A.: Lot sizing and scheduling with sequence dependent setup costs and times and efficient rescheduling opportunities. Int. J. Prod. Econ. **66**, 159–169 (2000)

ILOG: Cplex 7.1 User's manual. ILOG SA BP 85 9 Rue de Verdun 94253. Gentilly, France. http://www.ilog.com (2001)

Karimi, B., Fatemi Ghomi, S.M.T., Wilson, J.M.: The capacitated lot sizing problem: a review of models and algorithms. Omega **31**, 365–378 (2003)

Kelly, J.D., Mann, J.L.: Flowsheet decomposition heuristic for scheduling: a relax-and-fix method. Comput. Chem. Eng. **28**, 2193–2200 (2004)

Kuik, R., van Wassenhove, L.N., Maes, J.: Linear programming, simulated annealing and tabu search heuristics for lotsizing in bottleneck assembly systems. IIE Trans. **25**(1), 62–72 (1993)

Meyr, H.: Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. Eur. J. Oper. Res. **120**, 311–326 (2000)

Meyr, H.: Simultaneous lotsizing and scheduling on parallel machines. Eur. J. Oper. Res. **139**, 277–292 (2002)

Porkka, P., Vepsalainen, A.P.J., Kuula, M.: Multiperiod production planning carrying over set-up time. Int. J. Prod. Res. **41**, 1133–1148 (2003)

Ovacik, I.M., Uzsoy, R.: Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup time. Int. J. Prod. Res. **33**, 3173–3192 (1995)

Santos-Meza, E., Santos, M.O., Arenales, M.N.: Lot-sizing problem in an automated foundry. Eur. J. Oper. Res. **139**, 490–500 (2002)

Smith-Daniels, V.L., Ritzman, L.P.: A model for lot sizing and sequencing in process industries. Int. J. Prod. Res. **26**, 647–674 (1988)

Smith-Daniels, V.L., Smith-Daniels, D.E.: A mixed integer programming model for lot sizing and sequencing packaging lines in the process industries. IIE Trans. **18**, 278–285 (1986)

Stadtler, H.: Multilevel lot sizing with setup times and multiple constrained resources: internally rolling schedules with lot-sizing windows. Oper. Res. **51**, 487–502 (2003)

Suerie, C., Stadtler, H.: The capacitated lot-sizing problem with linked lot-sizes. Manag. Sci. **49**, 1039–1054 (2003)

Teghem, J., Pirlot, M., Antoniadis, C.: Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. J. Comput. Appl. Math. **64**, 91–102 (1995)

Wolsey, L.A., Integer Programming. Wiley, New York (1998)

Wolsey, L.A.: Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. Manag. Sci. **48**, 1587–1602 (2002)