INT. J. PROD. RES., 2000, VOL. 38, NO. 10, 2287–2307

# Rolling-horizon lot-sizing when set-up times are sequence-dependent

ALISTAIR R. CLARK[†]* and SIMON J. CLARK[‡]

The challenging problem of efficient lot sizing on parallel machines with sequence-dependent set-up times is modelled using a new mixed integer programming (MIP) formulation that permits multiple set-ups per planning period. The resulting model is generally too large to solve optimally and, given that it will be used on a rolling horizon basis with imperfect demand forecasts, approximate models that only generate exact schedules for the immediate periods are developed. Both static and rolling horizon snapshot tests are carried out. The approximate models are tested and found to be practical rolling horizon proxies for the exact model, reducing the dimensionality of the problem and allowing for faster solution by MIP and metaheuristic methods. However, for large problems the approximate models can also consume an impractical amount of computing time and so a rapid solution approach is presented to generate schedules by solving a succession of fast MIP models. Tests show that this approach is able to produce good solutions quickly.

## 1. Introduction

The simultaneous sizing and sequencing of production lots on a group of parallel machines is a problem that often occurs in industry at the level of tactical scheduling. Many companies manufacture a wide range of products or intermediate parts that share production capacity, but use up valuable production time when changing from one product to another. Examples witnessed by the authors in the auto, plastics and packaging industries include changing colours in printing or mixing machinery, and the time consuming swapping of templates in cutting or pressing shops. Although automation and process engineering has often substantially reduced the size of the problem, a surprisingly large number of companies still face substantial set-up times within an increasing range of products, with consequent losses of production capacity and missed deadlines if set-ups are not well managed and controlled.

While many production lots correspond to specific orders and so have a pre-determined size, a product or part may instead feed into many small distinct orders with different deadlines. In the packaging industry, for example, many unique orders may well use identical basic formats which can then be produced in one lot before printing customer-specific packaging. In such a situation, it makes sense to relate the product or part's lot-sizes to its total demand aggregated from the different orders. In other words, the problem becomes one of simultaneous sequencing and lot-sizing based on short-term forecasts of dependent demand.

Revision received June 1999.

† Faculty of Computer Studies and Mathematics, University of the West of England, Coldharbour Lane, Bristol, BS16 1QY, UK.

‡ Department of Economics, University of Edinburgh, UK.

* To whom correspondence should be addressed. e-mail: alistair.clark@uwe.ac.uk

Improved solutions to the management of set-ups can pay dividends in terms of more efficient use of capacity, better customer service and lower levels of processed inventory. It is, however, a difficult problem to solve optimally in the mathematical sense except for very small-sized instances. Its complexity necessitates the use of approximate solution methods that can rapidly generate good quality schedules that may well be sub-optimal, but which will be a substantial improvement on intuitive or indeed rule-generated schedules.

A huge amount of research has been carried out into lot-sizing. Recent surveys include Drexl and Kimms (1997) and Wolsey (1995). However, lot-sizing coupled with sequencing has been studied by far fewer researchers. The problem tackled in this paper includes representation of set-up times that are sequence-dependent and permit multiple set-ups within a planning period. It does not require that all or none of the production capacity on a machine be utilized within a period. As such, it is related to the capacitated lot-sizing problem (CLSP) as defined in Drexl and Kimms (1997), although the CLSP does not include sequencing decisions. Smith-Daniels and Smith-Daniels (1986) developed mixed integer programming (MIP) models and methods for the CLSP with sequence-dependent set-up times, but with the disadvantage that the solution methods were computationally feasible only for small problems. This paper seeks to overcome computational infeasibility for larger problems by substituting most of the integer 0/1 variables and constraints with continuous variables and constraints. Arosio and Sianesi (1993) also proposed a complex heuristic algorithm for simultaneous lot-sizing and sequencing on a single machine with sequence-dependent set-ups, but made no comparisons with optimal solutions in their computational tests. This paper does make such comparisons, albeit only for small problems.

Much of the research has concerned just the optimization of static lot-sizing or sequencing problems without taking into account the fact that the optimal solutions are usually applied on a rolling horizon basis. In this environment only the production decisions relating to the immediate periods are implemented after which the horizon is rolled forward and the model applied once more with updated demand, inventory and capacity information. A problem with the rolling horizon use of static lot-sizing models is that the demand in the new periods added as the horizon rolls forward can cause 'nervousness', i.e., changes in already planned but unimplemented lot-sizes may provoke additional set-ups, which a static solution applied in its entirety would avoid (Maes and Van Wassenhove 1986, Baker 1993, Kadipasaoglu and Sridharan 1997). This is particularly common for myopic planning horizons. Generally, as one would expect intuitively, researchers have found that the longer the planning horizon the better the rolling horizon performance of static models (Baker 1979), although this is not always the case (Baker 1977). However, short planning horizons need not be disadvantageous Maes and Van Wassenhove (1986) show that when the time between orders (TBO) is small, some myopic lot-sizing heuristics perform well, but will cause extra set-ups if the TBO is large.

Toklu and Wilson (1995) reported that a simple bottleneck heuristic they originally developed for static use resulted in a 50% cost increase when adapted for semi-myopic rolling horizon use, thus illustrating the perils of ignoring the reality of rolling horizons. Heuts *et al.* (1992) developed a heuristic for rolling-horizon lot-sizing and sequencing on a single machine with sequence-dependent set-up times. However, due to particularities of the chemical industry, they required that lot-sizes be integer multiples of a standard lot-size and included other special storage constraints, thus limiting its direct applicability. This paper models multiple machines in parallel with great flexibility as to when set-ups may occur.

Drexl and Kimms (1997) noted that little research has been carried out into the capacitated problem on a rolling horizon basis. In this category, Clark (1998) developed a very fast myopic rule-based heuristic for lot-sizing and sequencing on a set of parallel machines with sequence-dependent set-up times, the problem studied in this paper. It produced reasonable schedules that were robust to demand forecast errors. However, unlike the models developed in this paper, it assumed just a single set-up at the beginning of each period. Ovacik and Uzsoy (1995) presented a class of heuristic procedures to minimize maximum lateness on parallel identical machines with sequence dependent set-up times and dynamic job arrivals. They tried to steer a middle path between the high computational effort required by exact optimizing procedures on the one hand and the poor solution quality of myopic dispatching rules on the other hand. Ovacik and Uzsoy attempted this middle way with some success by breaking a large problem into a number of smaller semi-myopic ones that are solved exactly by branch and bound.

We use a similar, but different, approach in this paper by using branch-and-bound solutions for the immediate lot-sizing and sequencing production decisions that are implemented up to the next rolling horizon application of the model. The poor quality of myopic models is partly avoided by taking future demand and capacity into account. However this is achieved by not explicitly considering set-ups after the first period, but instead increasing unit production times to take likely set-ups into account.

The objective of this paper is to develop a practical rolling horizon method of determining efficient sequences and sizes of production lots on a set of parallel machines that have sequence-dependent set-up times. The method solves a succession of small MIPs with a tractable number of 0/1 variables. Nowadays, there are not only several fast high-quality solvers capable of tackling very large models on personal computers, but also powerful mathematical programming modelling languages such as XPRESS-MP (1997) and AMPL (Fourer *et al.* 1993) that permit rapid formulation and testing of models. It is the authors' experience in commercial projects that such tools are not only invaluable in the specification and prototyping of appropriate models, but also facilitate revisions as a client's environment and modelling needs change. As such, the authors have sought to solve the resulting flexible models by using generally applicable approaches that are not restricted to any specific problem, a point which is repeatedly cited by Maes and Van Wassenhove (1988) in support of mathematical programming based approaches.

The MIP approach developed in this paper is specifically designed to be applied solely on a rolling horizon basis. As such, several issues will be investigated in the computational tests:

- the quality of the approximating approach compared with the static optimal solutions of the exact model, particularly with respect to the implemented initial periods of the rolling horizon; this is possible only for small problems that can be solved optimally in feasible computing time;
- a comparison of the quality of schedules generated by successive rolling horizon applications of the approximating approach and the exact model;

- the number of products and machines that can be scheduled in reasonable computing time.

## 2. The exact model

To simplify notation, we assume that all products may be produced on any machine. However, a product may be restricted to certain machines without affecting the approaches presented in this paper. Let $i = 1, \dots, P$ denote the products, $m = 1, \dots, M$ the parallel machines, and $t = 1, \dots, T$ the demand periods. If up to $N$ set-ups per period are permitted then an exact mathematical formulation of the problem is the following mixed integer linear programme:

Model A:
$$\min \sum_{i,j} [h_i I_{it}^+ + g_i I_{it}^-] \tag{1}$$

such that

$$I_{i,t-1}^+ - I_{i,t-1}^- + \sum_{m,n} x_{imt}^n - I_{it}^+ + I_{it}^- = d_{it} \quad \forall i,t \tag{2}$$

$$\sum_j \sum_n \left[ \sum_i s_{ijm} y_{ijmt}^n + u_{jm} x_{jmt}^n \right] \leq A_{mt} \quad \forall m,t \tag{3}$$

$$y_{jim1}^1 = 0 \quad \forall i,\ m, j \neq j_{0m} \tag{4.0}$$

$$\sum_i y_{j_{0m}im1}^1 = 1 \quad \forall m \tag{4.1}$$

$$\sum_j y_{jimt}^n = \sum_k y_{ikmt}^{n+1} \quad \forall i,m,t,n = 1,\dots,N-1 \tag{4.n}$$

$$\sum_j y_{jim,t-1}^N = \sum_k y_{ikmt}^1 \quad \forall i,m,t = 2,\dots,T \tag{4.N}$$

$$x_{imt}^n \leq M_{imt} \sum_j y_{jimt}^n \quad \forall i,m,n,t \tag{5}$$

$$y_{jimt}^n = 0 \text{ or } 1 \quad \forall i,\ j,m,n,t \tag{6}$$

$$x_{imt}^n \geq 0 \quad \forall i,m,n,t \tag{7}$$

$$I_{it}^+ \geq 0; \quad I_{it}^- \geq 0 \quad \forall i,t \tag{8}$$

where the decision variables are as follows:

$$y_{ijmt}^n = \begin{cases} 1 & \text{if the } n\text{th setup on machine } m \text{ in period } t \text{ is from} \\ & \quad \text{product } i \text{ to product } j \\ 0 & \text{otherwise,} \end{cases}$$

$x_{imt}^n$ quantity of product $i$ produced between the $n$th and $(n+1)$th set-ups on machine $m$ in period $t$ (it is non-zero only if the $n$th set-up on machine $m$ is to product $i$),

| | |
|---|---|
| $I_{it}^{+}$ | stock of product $i$ at the end of period $t$, |
| $I_{it}^{-}$ | backlog of product $i$ at the end of period $t$. |

The parameters and data inputs are:

| | |
|---|---|
| $d_{it}$ | demand for product $i$ at the end of period $t$, |
| $A_{mt}$ | available time on machine $m$ in period $t$, |
| $s_{ijm}$ | time needed to set up from product $i$ to product $j$ on machine $m$, |
| $u_{im}$ | time needed to produce one unit of product $i$ on machine $m$, |
| $h_i$ | cost of holding one unit of product $i$ from one period to the next, |
| $g_i$ | penalty cost of carrying over a backorder of one unit of product $i$ from one period to the next, |
| $j_{0m}$ | the product produced on machine $m$ at the end of period 0, i.e. the starting set-up configuration on machine $m$. |

and where

$$M_{imt} = \min\left\{ A_{mt}/u_{im},\ \sum_{t=1}^{T} d_{it} + I_{i0}^{-} - I_{i0}^{+} \right\} \tag{9}$$

is an upper bound on $x_{imt}^{n}$, since all backlogs and future production of product $i$ might, in theory, be produced on machine $m$. Model A has $PM(PNT - P + 1)$ 0/1 variables and $PMNT$ continuous variables.

Note that, unlike many formulations in the literature, model A allows backlogs. To prohibit them is unrealistic—many companies face occasional or frequent capacity overloads and they often have no immediate choice but to backlog demand. The inclusion of the possibility of backlogs is necessary in useful models—a MIP solver output message of 'No feasible solution exists' is not helpful to the user when backlogs are not represented in the model but do in fact occur in reality. The question of what penalties, $g_i$, should be allocated to backlogs can only be answered for each individual context, depending on market conditions and the importance of the customers for a particular product. Since the value of such penalties will partly be based on judgement and imprecise information, there is little added value in investing the often huge extra effort needed to solve the model optimally rather than approximately. Only holding and backlog penalty costs are included in the objective function (1) since these are our major concerns. We assume that the total costs of the provision of the production capacity $A_{mt}$ are fixed and do not depend on the production decision variables $x_{imt}^{n}$ and $y_{ijmt}^{n}$. Additional set-up and direct production costs are not included in the objective function as they are likely to vary little or be negligible in comparison with the penalty costs of the additional backlogs provoked by the lost machine time that an inefficient sequence of set-ups would cause. However, if need be, such costs can be incorporated into the objective function without difficulty by the inclusion of a term such as

$$\sum_{j}\sum_{t}\sum_{n}\left[\sum_{i} SetupCost_{ijm} y_{ijmt}^{n} + UnitCost_{jm} x_{jmt}^{n}\right].$$

Constraints (2) are the standard equations linking inventory, production and demand while constraints (3) represent the limited availability of capacity. Note that since the machine time capacity parameter $A_{mt}$ is indexed on $t$, the production periods $t$ in the model may be of different lengths. For example, weekend working may be combined into a single planning period. Constraints (4) ensure that a set-up on a machine must, and may only, occur between a single pair of products, possibly both the same product, and that if a certain product is changed to, then it must be changed from in the following set-up. The equals sign $=$, rather than the $\leq$ sign, is necessary in constraints (4) so that we always know for which product a machine is configured, especially when it is not producing. Thus, the combination $y^n_{jimt} = 1$ and $x^n_{imt} = 0$ must be allowed. Note that constraints (4) require that for each triple $(n, m, t)$ there is exactly one pair $(i, j)$ for which $y^n_{jimt} = 1$, i.e. there must be precisely $N$ set-ups in each period on each machine, even if a set-up $y^n_{iimt} = 1$ is just from a product $i$ to itself. Since a set-up time $s_{iim}$ from a product $i$ to itself is zero, note that the model does not force a machine to have exactly $N$ positive-time set-ups but rather up to $N$ such set-ups. The remaining zero-time set-ups are modelling phantoms and do not exist in reality. Constraints (5) ensure that there must be a set-up if a product is produced on a machine in a period, even if it is just a phantom one from a product to itself. Wolsey (1997) points out that network flow type constraints such as (4) produce a very tight formulation in that the relaxed solution is the integer solution when no other constraints are present other than (6). The hope is that constraints (4) will assist a branch-and-bound search to converge more rapidly.

The first set-up in a period on a machine must occur at the beginning of the period, but the subsequent $N - 1$ set-ups may occur at any time within the period. If the constraints

$$y^1_{iimt} = 1 \quad \forall i, m, t \quad \text{and} \quad y^1_{jimt} = 0 \quad \forall i, j, m, t | i \neq j \tag{10}$$

are imposed, then model A is restricted to just $N - 1$ set-ups per period, but these may occur at any time within the period. Thus, the model is related to (and more general than) the proportional lot-sizing and scheduling problem (PLSP), which allows the single permitted set-up in each period to occur at any time within the period, if at all (Drexl and Haase 1995, Drexl and Kimms 1997).

## 3. The approximate rolling horizon models

Since, in practice, the backorder penalties, $g_i$, are usually subjective estimates and, moreover, the demand forecasts $d_{i2}, \ldots, d_{iT}$ are invariably updated as the planning horizon is rolled forward, there is limited added benefit in investing substantial computing effort in attempting to solve model A optimally if, instead, we can quickly arrive at a near-optimal solution, especially concerning the first few periods. Plant schedulers far prefer models that make decisions quickly, since many scenarios (usually concerning demand forecasts and machine availability) often have to be evaluated with what-if analyses before a final decision is made. To achieve such speed, two approximations are proposed, one to the model, and one to the solution method.

In a rolling horizon environment, only the decisions relating to the first $\tau$ periods are implemented before advancing the horizon by $\tau$ periods and applying the model again. This gives rise to the question: is it necessary to have 0/1 solution values for $y^n_{jimt}$ relating to the set-ups in the last $T - \tau$ periods? Why not reduce the number of 0/1 variables by representing the set-ups for the last $T - \tau$ periods as 'relaxed' continuous variables between 0 and 1? The payoff would be that solving just the first $\tau$ periods' $y^m_{jimt}$ 0/1 variables and relaxing the rest saves substantial computing time. The appropriate model, denominated B, is obtained by replacing the constraints $(6.t)$ with the constraints:

$$y^n_{jimt} = 0 \text{ or } 1 \quad \forall i,\ j, m, n, t = 1, \dots, \tau \tag{6.$\tau$}$$

$$0 \le y^n_{jimt} \le 1 \quad \forall i,\ j, m, n, t = \tau + 1, \dots, T \tag{6.$T$}$$

Even more computing time is economized by eliminating the $y^n_{jimt}$ variables from the last $T - \tau$ periods. Their removal is compensated for by increasing the values of the unit production times $u_{imt}$ to take the set-up times $s_{jimt}$ into account. The following model accomplishes this:

model C:
$$\min \sum_{i,t} [h_i I^+_{it} + g_i I^-_{it}] \tag{1}$$

such that

$$I^+_{i,i-1} - I^-_{i,t-1} + \sum_{m,n} x^n_{imt} - I^+_{it} + I^-_{it} = d_{it} \quad \forall i, t = 1, \dots, \tau \tag{2.$\tau$}$$

$$I^+_{i,t-1} - I^-_{i,t-1} + \sum_{m} x^{\mathrm{s}}_{imt} - I^+_{it} + I^-_{it} = d_{it} \quad \forall i, t = \tau + 1, \dots, T \tag{2.$t$}$$

$$\sum_j \sum_n \left[ \sum_i s_{ijm} y^n_{ijm1} + u_{jm} x^n_{jm1} \right] \le A_{m1} \quad \forall m \tag{3.1}$$

$$\sum_{j,n} \left[ \sum_i s_{ijm} y^n_{ijmt} + u_{jm} x^n_{jmt} \right] \le A_{mt} \quad \forall m, t = 2, \dots, \tau \tag{11.$\tau$}$$

$$\sum_i u^{\mathrm{s}}_{im} x^{\mathrm{s}}_{imt} \le A_{mt} \quad \forall m, t = \tau + 1, \dots T \tag{11.$t$}$$

$$y^1_{jim1} = 0 \quad \forall i, m,\ j \ne j_{0m} \tag{4.0}$$

$$\sum_i y^1_{j_{0m} im1} = 1 \quad \forall m \tag{4.1}$$

$$\sum_j y^n_{jimt} = \sum_k y^{n+1}_{ikmt} \quad \forall i, m, n = 1, \dots, N-1, t = 1, \dots, \tau \tag{12.$n$}$$

$$\sum_j y^N_{jim,t-1} = \sum_k y^1_{ikmt} \quad \forall i, m, t = 2, \dots, \tau \tag{12.$N$}$$

$$x^n_{imt} \le M_{imt} \sum_j y^n_{jimt} \quad \forall i, m, n, t = 1, \dots, \tau \tag{13}$$

$$y^n_{jimt} = 0 \text{ or } 1 \quad \forall i,\ j, m, n, t = 1, \dots, \tau \tag{14}$$

$$x^n_{imt} \ge 0 \quad \forall i, m, n, t = 1, \dots, \tau \tag{15}$$

$$x^{\mathrm{s}}_{imt} \ge 0 \quad \forall i, m, t = \tau + 1, \dots, T \tag{16}$$

$$I_{it}^{+} \geq 0; \quad I_{it}^{-} \geq 0 \quad \forall i, t \tag{8}$$

where

| | |
|---|---|
| $x_{imt}^{s}$ | is the production of product $i$ on machine $m$ in period $t = \tau + 1, \ldots, T$, |
| $u_{im}^{s}$ | is the estimated increased production time needed to produce one unit of product $i$ on machine $m$, taking possible set-up time into account. |

The aim of using a linear programme (LP) to model production in periods $\tau + 1, \ldots, T$ is to simulate the levels of stocks $I_{i\tau}^{+}$ and backorders $I_{i\tau}^{-}$ at the end of period $\tau$, as these define the boundary between the MIP of periods 1 to $\tau$ and the LP. If these stock and backorder levels were perfectly simulated then, in a rolling horizon environment model, C would exactly reproduce the results of model A. We cannot hope for a perfect simulation, but we can try to estimate the MIP/LP boundary stock and backorder levels by approximately replicating the overall consumption of production and set-up times in periods $\tau + 1, \ldots, T$ via the increased unit production time $u_{imt}^{s}$. The difficulty with this approach is that we have to conjecture the form of an optimal sequence of lots in order to calculate reasonable values for $u_{imt}^{s}$.

How then should $u_{imt}^{s}$ be calculated? Suppose the optimal policy were to meet the demand of every period on a lot-for-lot basis, with no between-periods stocks. This would imply producing, at best, $P$ lots on the $M$ machines, i.e. $P/M$ set-ups per machine in each of periods $\tau + 1, \ldots, T$. Product $i$ would have an average lot-size of $\bar{d}_i$, the mean demand of product $i$. As a result, the increased unit production time in constraint $(11.t)$ could be estimated as

$$u_{im}^{s} = \left(\bar{s}_{im}^{\text{to}} + u_{im}\bar{d}_i\right)/\bar{d}_i \tag{17}$$

where $\bar{s}_{im}^{\text{to}}$ is the mean set-up time to product $i$ on machine $m$. However, if the optimal policy were to produce lots less frequently than lot-for-lot, then expression (17) would overestimate $u_{im}^{s}$ and possibly cause over-production in periods 1 to $\tau$, resulting in more inventory than necessary, but reducing the possibility of backlogs. If an optimal policy were to produce more frequently than lot-for-lot, then expression (17) would underestimate $u_{im}^{s}$ and possibly not cause a necessary anticipation of production in periods 1 to $\tau$ of a demand peak in periods $\tau + 1$ to $T$. Since demand forecasts generally change as the planning horizon rolls forward, such production in periods 1 to $\tau$ might, in fact, not be needed. Moreover, greater importance is usually attached to avoiding backlogs than to reducing inventory. Accordingly, expression (17) seems a reasonable choice as a practical estimator of $u_{im}^{s}$.

The manufacture of a product implies that time will be spent on set-ups *from* that product as well as *to* the product. For example, a dark coloured ink in printing machinery will tend to have larger set-up-from times than set-up-to times. Thus, as an alternative to expression (17), $u_{im}^{s}$ could be estimated using the mean set-up time $\bar{s}_{im}^{\text{from}}$ from product $i$ as well as the mean set-up time $\bar{s}_{im}^{\text{to}}$ to product $i$ on machine $m$:

$$u_{im}^{s} = \left(0.5\bar{s}_{im}^{\text{from}} + 0.5\bar{s}_{im}^{\text{to}} + u_{im}\bar{d}_i\right)/\bar{d}_i \tag{18}$$

This alteration results in a new model called D.

The permitted number $N$ of set-ups per period that models C and D may schedule is under the user's control. The effect of different values of $N$ for different combinations of $P$ and $M$ will be explored in the computational tests, but the user should *a priori* estimate from his or her knowledge of the situation what would be the likely value of $N$

in an optimal plan. It seems intuitively sensible that the larger the ratio $P/M$ the larger the number, $N$, of set-ups that it is desirable to allow. A natural lot-for-lot choice for $N$ would seem to be $N = P/M$. For smaller (greater) values of $N$, there will be fewer (more) set-ups than $P/M$, and so the total amount of set-up time could be reduced (increased) accordingly, suggesting the following modified expressions for $u^{\mathrm{s}}_{im}$:

$$u^{\mathrm{s}}_{im} = \begin{cases} (\bar{s}_{im}NM/P + u_{im}\bar{d}_i)/\bar{d}_i & \text{in model C} \qquad (19) \\ ((0.5\bar{s}^{\mathrm{from}}_{im} + 0.5\bar{s}^{\mathrm{to}}_{im})NM/P + u_{im}\bar{d}_i & \text{in model D.} \qquad (20) \end{cases}$$

These two adjustments result in new models called E and F respectively.

We are interested in the following question: what is the difference in the value of the objective function between the optimal solutions of model A and models C to F? In other words, how good an approximation is each of models C to F to model A? (Since B is a relaxation of A, then its optimal solution will necessarily be less than that of A). In particular, since all the models are applied on a rolling horizon basis every $\tau$ periods, how different are their optimal solutions over the first $\tau$ periods? This question can only be answered for problems that are small enough for model A to be solved optimally.

All six models were tested for $T = 5$ periods with known future demand. Models B to F were applied with $\tau = 1$. The unit stockholding cost of product $i$ is $h_i = 1.0$ units of value per period, and the backlog cost $g_i = 100h_i$, so that great emphasis is attached to avoiding backlogs, but they are not prohibited. The set-up times are generated with mean $\bar{s} = 1.0$ hours in such a manner that if $i < j$ then $s_{ijm} < 1$ and $s_{jim} > 1$, simulating, for example, set-ups between light $(i)$ and dark $(j)$ colours in printing machines. The set-up times also obey the triangle rule, i.e. it is always quicker to change directly from one product to another than to do so via an intermediate product. The unit production times $u_{im}$ are distributed U(0.005, 0.015) with mean $\bar{u} = 0.01$ hours. The tests are carried out under conditions of tight capacity, which is defined as allowing for no set-ups. This may result in backlogs, but the objective of the tests is to evaluate how well the various models manage to reduce backlogs. A product's mean demand per period $\bar{d}$ under tight capacity is thus calculated as:

$$\bar{d} = \frac{M\bar{A}}{\bar{u}P}, \tag{21}$$

where $\bar{A} = 24$ hours per period, i.e. all products have the same mean demand. The demand $d_{it}$ is distributed U$(0.5\bar{d}, 1.5\bar{d})$.

The measure of performance is the value of the objective function over a horizon of five periods:

$$\sum_i \sum_{t=1}^{5} [h_i I^+_{it} + g_i I^-_{it}]. \tag{22}$$

*A priori*, we expect a greater number of set-ups per period to provide more flexibility and it will be interesting to see how large is the difference in performance between the number $N$ of set-ups allowed per period. The computational tests were conducted on a Sun Sparc 10 workstation with the SunOS 4.1.4 operating system and 128 Mb of RAM, using the C++ programming language and the XPRESS-MP Workstation Model Builder Release 10.02 with Integer Optimizer Release 10.07 (1997).

As optimal solutions will only be found within a tolerable amount of computing time for small-sized MIPs, model A was initially solved for a set of randomly generated problems with $M = 2$ machines and $P = 2$ to 6 products. Each problem was solved with $N = 1$ to 6 set-ups per period, for a maximum of 24 hours CPU time, obtaining values for the objective function over all 5 periods and the total CPU time in minutes, as shown in table 1 (where the less-than-or-equal symbol $\leq$ indicates the incumbent solution after 24 hours CPU time rather than the optimal solution).

Note that Model A is solvable within reasonable time when just one set-up per period is permitted. It is interesting that permitting two set-ups per period provides a flexibility that immediately reduces the objective function, even with just two products. However, higher values of $N$ for two products do not yield better results and generally take longer to solve, albeit in well under a minute. Permitting two or more set-ups per period for 3 and 4 products permits the total elimination of stock and backorders. It is with 5 and 6 products that the problem of combinatorial explosion causes the $N = 2$ and 5 models to need more than 24 hours to be solved optimally.

This snapshot sample verifies that the modelling of multiple set-ups per period enables a much lower level of backorders and stocks, as would be expected theoretically, confirming the usefulness of Model A. However it also suggests that comparisons of optimal solutions with Models B to F can only be carried out with, at most, five products. A problem was randomly generated for each of $P = 4$ to 6 products, each problem then being solved with $N = 1$ to 6 set-ups per period by models A to C. The results are given in tables 2 to 4. Besides the value of the objective function over all five periods (Obj. Fn. 5) and the total CPU time in

| Objective Function<br>*CPU minutes* | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ | $N = 6$ |
|---|---|---|---|---|---|---|
| $P = 2$ products | 80 719 | 76 412 | 76 412 | 76 412 | 76 412 | 76 412 |
| | *0.00* | *0.00* | *0.00* | *0.083* | *0.0167* | *0.2* |
| $P = 3$ products | 105 839 | 0 | 0 | 0 | 0 | 0 |
| | *0.067* | *0.0167* | *0.0167* | *0.933* | *2.083* | *0.133* |
| $P = 4$ products | 190 826 | 0 | 0 | 0 | 0 | 0 |
| | *1.13* | *4.25* | *0.03* | *0.05* | *0.03* | *0.03* |
| $P = 5$ products | 379 760 | $\leq$ 51 667 | 0 | 0 | $\leq$ 13 | 0 |
| | *5.5* | *>1440* | *942.9* | *775.2* | *>1440* | *96.77* |
| $P = 6$ products | 632 115 | $\leq$ 96 164 | $\leq$ 25 740 | $\leq$ 6 994 | $\leq$ 24 203 | $\leq$ 13 997 |
| | *19.7* | *>1440* | *> 1440* | *>1440* | *>1440* | *>1440* |

Table 1. Solutions to model A for two machines.

| Obj. Fn. 1<br>Obj. Fn. 5<br>*CPU minutes* | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ | $N = 6$ |
|---|---|---|---|---|---|---|
| | 184 737 | 0 | 0 | 0 | 0 | 0 |
| A | 190 826 | 0 | 0 | 0 | 0 | 0 |
| | *1.13* | *4.25* | *0.03* | *0.05* | *0.03* | *0.03* |
| | 182 419 | 0 | 0 | 0 | 0 | 0 |
| B to F | 182 419 | 0 | 0 | 0 | 0 | 0 |
| | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* |

Table 2. Solutions to models A to D for four products on two machines.

| Obj. Fn. 1<br>Obj. Fn. 5<br>*CPU minutes* | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ | $N = 6$ |
|---|---|---|---|---|---|---|
|  | 246 136 | $\leq$ 49 321 | 0 | 0 | 0 | 0 |
| A | 379 760 | $\leq$ 51 667 | 0 | 0 | $\leq$ 13 | 0 |
|  | *5.47* | *>1440* | *943* | *775* | *>1440* | *97* |
|  | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
| B | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
|  | *0.00* | *0.22* | *0.03* | *0.00* | *0.00* | *0.38* |
|  | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
| C | 195 140 | 49 204 | 384 | 384 | 384 | 384 |
|  | *0.00* | *0.07* | *0.00* | *0.02* | *0.02* | *0.08* |
|  | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
| D | 195 152 | 49 217 | 396 | 396 | 396 | 396 |
|  | *0.00* | *0.03* | *0.00* | *0.00* | *0.02* | *0.02* |
|  | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
| E | 194 756 | 49 083 | 497 | 699 | 1131 | 1617 |
|  | *0.00* |  | *0.00* | *0.00* | *0.03* | *0.03* |
|  | 194 756 | 48 821 | 0 | 0 | 0 | 0 |
| F | 194 756 | 49 087 | 521 | 747 | 1196 | 1746 |
|  | *0.05* | *0.03* | *0.03* | *0.00* | *0.00* | *0.10* |

Table 3. Solutions to Models A to D for five products on two machines.

| Obj. Fn. 1<br>Obj. Fn. 5<br>*CPU minutes* | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ | $N = 6$ |
|---|---|---|---|---|---|---|
|  | 284 766 | $\leq$ 92 009 | $\leq$ 24 064 | $\leq$ 5945 | $\leq$ 11 251 | $\leq$ 12 643 |
| A | 632 115 | $\leq$ 96 163 | $\leq$ 25 740 | $\leq$ 6994 | $\leq$ 24 203 | $\leq$ 13 997 |
|  | *19.7* | *>1440* | *>1440* | *>1440* | *>1440* | *>1440* |
|  | 282 592 | 90 645 | 10 198 | 0 | 0 | 0 |
| B | 282 592 | 90 645 | 10 198 | 0 | 0 | 0 |
|  | *0.05* | *0.63* | *4.42* | *21.55* | *19.35* | *130.20* |
|  | 283 402 | 90 645 | 10 198 | 0 | 0 | 0 |
| C | 284 104 | 91 346 | 10 900 | 702 | 702 | 702 |
|  | *0.00* | *0.15* | *1.78* | *5.48* | *39.95* | *30.00* |
|  | 283 515 | 90 693 | 10 198 | 0 | 0 | 0 |
| D | 287 367 | 91 457 | 10 962 | 764 | 764 | 764 |
|  | *0.00* | *0.15* | *2.00* | *6.25* | *47.35* | *37.92* |
|  | 282 705 | 90 645 | 10 198 | 0 | 112 | 113 |
| E | 282 852 | 90 966 | 10 900 | 1126 | 1635 | 41 495 |
|  | *0.00* | *0.15* | *1.77* | *7.33* | *48.23* | *5.40* |
|  | 282 741 | 90 645 | 10 198 | 0 | 1130 | 10 576 |
| F | 282 897 | 90 989 | 10 962 | 1234 | 5822 | 56 224 |
|  |  | *0.15* | *2.00* | *10.58* | *11.57* | *3.55* |

Table 4. Solutions to models A to D for six products on two machines.

seconds, the value of the objective function in just the first period (Obj. Fn. 1) was also noted as this is the only part of the solution that will be used when the models are implemented on a rolling horizon basis with $\tau = 1$.

Note from table 2 that models B to F underestimate the optimal solution of model A by over 1% when there are four products, but run virtually instantly compared with model A, which took over 4 minutes to identify the zero optimal solution for two set-ups per period. It is interesting that it does not necessarily take longer to solve the models with larger values of $N$.

Observe in table 3 that model A with five products is starting to become too large to solve optimally within 24 hours CPU time when more than two set-ups per period are permitted, while models B to F are solved in a fraction of a minute. When just one set-up is permitted, models B to F underestimate the optimal solution by 21% for period 1 and by 49% for all five periods. However, since the ratio $P/M$ of products to machines is $2\frac{1}{2}$, we would expect the best production plan to have two or three set-ups per period. This is born out by the results for $N = 2$, where model A substantially reduced its solution value (but needed over 24 hours CPU time to do so). Models B to F are much better approximations when two or more set-ups are permitted.

In table 4, model A with six products is now too large to solve optimally within 24 hours CPU time when more than two set-ups per period are permitted. When just one set-up is permitted, models B to F underestimate the optimal solution by less than 1% for period 1 and by 55% for all five periods. The models appear to be a much better approximation over the whole horizon when two set-ups or more are permitted, but we cannot know for sure as model A's optimal solution was not identified. When three set-ups are permitted, models B to F identified a zero solution for the first periods, whereas when four or more are permitted, only models B to D managed to identify a zero solution. In fact, the solutions for models E and F progressively worsened for five and six set-ups, a pattern that was repeated in other problems reported below.

Further tests showed that, in comparison to model B, models C to F provide solutions with equal or very slightly higher objective function values but within a small fraction of model B's CPU time. As a result, we discard model B from now on.

The tests above are encouraging and suggest that the period 1 solution of model A is well approximated by models C to F. However, the results of tables 3 and 4 raise two concerns, namely:

(1) that the linear approximations of models C to F are underestimating the backorders in periods 2 to 5, probably due to the rigidity of the condition in model A that only $N$ set-ups at most are permitted in each of these periods.
(2) that as the value of the objective function in period 1 in the solutions to model A is only slightly less than its value over all 5 periods, the backorders in the first period are being caused by the zero initial starting stocks rather than by the supposedly tight capacity due to expression (21); this could be investigated by comparison of results using both the original demand as defined by expression (21) and demand increased by 50% so as to cause more backorders;

If models C to F tend to underestimate the unit capacity needed for production in periods $\tau + 1$ onwards, and demand is high relative to actual capacity, then the model will provisionally plan for these periods production, which in fact it should bring forward in time and produce in periods 1 to $\tau$. When the horizon is then rolled forward by $\tau$ periods, there might not be sufficient capacity to produce all the demand in the new periods 1 to $\tau$, causing backorders and a higher value of the objective function. Static tests will not detect this. Rolling horizon tests must be carried out, as in the next section.

## 4. Rolling horizon tests

What kind of performance in minimizing backorders and levels of finished stocks can we expect from models A and C to F over a rolling horizon? To gain some insight, computational tests were carried out on a randomly generated problem for each of certain combinations of $M$ and $P$, using the same parameters as in the static tests above. A horizon of $T = 5$ periods was used, applied with $\tau = 1$ period over ten rolling horizons with known future demand. The measure of performance was the value of the objective function:

$$\sum_{i} \sum_{t=1}^{10} [h_i I_{it}^{+} + g_i I_{it}^{-}] \tag{23}$$

resulting from the production actually implemented in each of the ten periods. Each problem was solved for:

(1) each of models A and C to F when computationally practicable,
(2) both the original demand, as defined by expression (18), and the same demand increased by 50%,
(3) a range of values of $N$.

Tables 5 and 6 show the results of initial tests on the largest problem for which model A could be solved optimally in reasonable CPU time, namely, four products

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| A | 206 191 | 1293 | 0 | 0 | 0 |
| | *9.03* | *28.43* | *2.05* | *1.13* | *2.15* |
| C | 1 870 406 | 0 | 0 | 0 | 0 |
| | *0.00* | *0.02* | *0.00* | *0.00* | *0.02* |
| D | 1 854 634 | 0 | 0 | 0 | 0 |
| | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* |
| E | 2 005 569 | 0 | 0 | 129 | 554 |
| | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* |
| F | 1 999 272 | 0 | 0 | 154 | 629 |
| | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* |

Table 5. Rolling horizon solutions for four products on two machines, with the original demand.

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| A | 1 674 997 | 956 142 | not known | n.k. | n.k. |
| | *2.18* | *324.12* | *> 2100* | too long | too long |
| C | 1 881 455 | 630 028 | 717 758 | 717 758 | 717 758 |
| | *0.00* | *0.00* | *0.00* | *0.47* | *1.45* |
| D | 1 890 426 | 611 277 | 711 811 | 711 811 | 711 811 |
| | *0.00* | *0.00* | *0.08* | *0.30* | *0.90* |
| E | 2 345 155 | 630 028 | 570 182 | 548 447 | 704 957 |
| | *0.00* | *0.00* | *0.03* | *0.23* | *0.43* |
| F | 2 354 399 | 611 277 | 571 092 | 571 092 | 549 357 |
| | *0.00* | *0.00* | *0.02* | *0.17* | *0.43* |

Table 6. Rolling horizon solutions for four products on two machines, with 50% extra demand.

on two machines. The first figure shows the value of expression (23) while second figure shows the total CPU time in minutes over all ten rolling horizons.

Commenting on the results of table 5, we see that model A gave a far superior solution than the other models for $N = 1$, but at the expense of substantial CPU time. However, since the ratio $P/M$ of products to machines is 2, we would expect production plans constrained by $N = 1$ permitted set-ups per period to result in an inferior solution for any of models A to F. This expectation is born out by the results when $N = 2$, where model A substantially reduced its solution value (but needed even greater CPU time to do so). Models C to F all did even better, obtaining a zero value solution in just one second. This may seem contradictory at first sight, given that A is an optimal static model, but keep in mind that here all the models are being applied dynamically on a rolling horizon basis with only the first period of each MIP contributing to the ten-period solution value in expression (23). Rolling horizon outcomes can be very different from static ones, as mentioned in the introduction. When $N = 3$, all five models produced zero solutions, although model A needed two minutes' CPU time to do so. For $N = 4$ and 5, models C and D did well, particularly the latter, outperforming E and F. Since we are dealing with rolling horizons and not a single static horizon, it is quite possible for the solution of models E and F to worsen in the passage from the $N = 3$ model formulation to the statically more flexible $N = 4$ formulation and then to $N = 5$.

This behaviour bodes well for models C to F. However, is it repeated when the same demand is increased by 50%? Table 6 shows that, not surprisingly, there are now no zero solutions as backorders are inevitable. For $N = 1$, model A did better than models C and D which, in turn, gave noticeably better solutions than models E and F. However, when $N = 2$, models C to F all gave better solutions than model A and in virtually no time at all. In contrast, model A took over 5 hours CPU time and, in fact, just the first MIP of model A took over an hour to solve. For $N = 3$, a feasible solution to the first MIP of model A had not even been identified after 35 hours CPU time. Models C and D provide worse solutions than for $N = 2$, with a steady state being reached with no improvement for higher values of $N$. Model E's solution significantly worsened for $N = 5$. Overall, it appears that $N = 2$ or 3 gave the most reasonable solutions with the least CPU time.

Tables 7 and 8 show the experimental results of similar tests for six products on two machines, excluding model A which cannot be solved optimally for $N > 1$, for problems larger than four products. For the original demand, solutions improve as $N$ increases up to 4, with models C and D generally doing much better than models E

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 1 711 167 | 483 343 | 18 210 | 7236 | 8513 |
| | *0.00* | *1.00* | *13.05* | *52.77* | *232.15* |
| D | 1 667 553 | 474 530 | 17 904 | 8110 | 7628 |
| | *0.00* | *1.00* | *6.35* | *34.42* | *188.85* |
| E | 2 382 416 | 653 467 | 18 210 | 11 684 | 19 459 |
| | *0.00* | *1.03* | *13.05* | *30.23* | *151.37* |
| F | 2 386 068 | 587 592 | 17 904 | 12 640 | 17 343 |
| | *0.00* | *1.03* | *6.18* | *30.35* | *265.78* |

Table 7. Rolling horizon solutions for six products on two machines, with the original demand.

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 8 488 334 | 7 299 680 | 7 162 963 | 7 162 963 | 7 162 963 |
| | *0.00* | *0.03* | *0.30* | *1.27* | *3.58* |
| D | 8 462 219 | 7 464 454 | 7 163 013 | 7 163 013 | 7 563 781 |
| | *0.00* | *0.02* | *0.35* | *1.28* | *3.60* |
| E | 8 301 999 | 7 299 666 | 7 162 963 | 7 162 963 | 7 199 068 |
| | *0.00* | *0.02* | *0.30* | *1.22* | *3.58* |
| F | 8 275 882 | 7 410 722 | 7 163 013 | 7 163 013 | 7 163 013 |
| | *0.00* | *0.03* | *0.33* | *1.20* | *3.57* |

Table 8. Rolling horizon solutions for six products on two machines, with 50% extra demand.

and F. The solution times increase with $N$, and are really only practicable for up to $N = 4$ which also gives the best solution. When demand is increased by 50%, CPU times become longer as $N$ increases with solutions improving and then becoming stationary at $N = 3$ and after. Overall, the most reasonable solutions with the least CPU time were obtained with $N = 3$ or 4.

Tables 9 and 10 show the results for six products on three machines. The best value of $N$ appears to be 3, obtaining the best solution with the least CPU time. Models C and D generally do better than models E and F, obtaining an optimal zero solution with the original demand when three or more set-ups are permitted. Tables 11 and 12 show a very similar pattern of results for five products on five machines. The best value of $N$ appears to be 2, in that it obtains the best solution with the least CPU time. Once again, models C and D generally do better than models E and F,

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 2 253 051 | 13 389 | 0 | 0 | 0 |
| | *0.03* | *5.55* | *2.67* | *6.05* | *203.55* |
| D | 2 169 981 | 15 031 | 0 | 0 | 0 |
| | *0.03* | *8.47* | *0.95* | *11.28* | *66.62* |
| E | 2 468 314 | 13 389 | 287 | 2007 | 5878 |
| | *0.03* | *5.55* | *1.48* | *3.98* | *14.58* |
| F | 2 363 854 | 15 031 | 319 | 2166 | 6160 |
| | *0.05* | *8.47* | *0.02* | *4.60* | *229.92* |

Table 9. Rolling horizon solutions for six products on three machines, with the original demand.

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 10 863 189 | 9 406 345 | 9 370 071 | 9 370 071 | 9 370 071 |
| | *0.00* | *0.10* | *1.95* | *7.97* | *23.28* |
| D | 10 970 045 | 9 385 619 | 9 341 367 | 9 341 367 | 9 341 367 |
| | *0.00* | *0.17* | *1.88* | *8.72* | *25.05* |
| E | 10 970 045 | 9 406 345 | 9 263 607 | 9 290 062 | 9 447 347 |
| | *0.00* | *0.12* | *1.13* | *7.28* | *14.53* |
| F | 10 970 045 | 9 385 619 | 9 366 834 | 9 284 958 | 9 279 843 |
| | *0.00* | *0.17* | *1.75* | *5.45* | *21.43* |

Table 10. Rolling horizon solutions for six products on three machines, with 50% extra demand.

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 219 384 | 0 | 0 | 0 | 0 |
| | *0.12* | *0.25* | *0.80* | *3.03* | *9.43* |
| D | 217 729 | 0 | 0 | 0 | 0 |
| | *0.20* | *0.47* | *0.88* | *0.65* | *8.70* |
| E | 219 384 | 0 | 605 | 2169 | 3861 |
| | *0.12* | *0.55* | *0.62* | *0.92* | *2.72* |
| F | 217 729 | 0 | 633 | 2193 | 3733 |
| | *0.18* | *1.13* | *0.97* | *0.95* | *0.58* |

Table 11. Rolling horizon solutions for five products on five machines, with the original demand

| Model | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|
| C | 3 242 701 | 2 259 824 | 2 155 196 | 2 155 196 | not known |
| | *0.00* | *0.15* | *7.07* | *156.75* | *> 600* |
| D | 3 243 539 | 2 161 323 | 2 161 855 | 2 161 855 | n.k. |
| | *0.00* | *0.13* | *6.47* | *154.101* | n.k. |
| E | 3 242 701 | 2 302 284 | 2 401 327 | 2 476 528 | n.k. |
| | *0.00* | *0.13* | *6.60* | *189.98* | n.k. |
| F | 3 243 539 | 2 240 733 | 2 337 509 | 2 564 445 | n.k. |
| | *0.00* | *0.13* | *6.63* | *120.32* | n.k. |

Table 12. Rolling horizon solutions for five products on five machines, with 50% extra demand.

obtaining an optimal zero solution with the original demand when two or more set-ups are permitted.

The results of Tables 5 to 12 and additional tests, although a snapshot, suggest that, in a rolling horizon environment, models C and D are better proxies for model A than models E and F. It also appears that a value of $N = (P/M + 1)$ or $P/M$ gives the best solution with least computing effort. However, it is clear that models C or D will take too long to solve optimally for only slightly larger problems than those considered above. This concern motivated the development of faster but sub-optimal solution methods in the next section.

## 5. The fix-&-relax solution approach

Model C has $PM(N\tau + T - \tau)$ continuous variables and $PM(PN\tau - P + 1)$ 0/1 variables and, as verified in the test above, can only be solved optimally for small-sized problems since the time taken to solve a MIP tends to explode exponentially as the number of 0/1 variables increases. On the grounds that it is better to have a good solution in one's hand than insist on an unobtainable optimal solution, we could simply place an upper limit on the time spent solving a MIP in model C. However, this is somewhat arbitrary, depending on the default branch-and-bound search strategy used in the particular MIP solver. There is also no guarantee that even a feasible solution, let alone a satisfactory one, will be identified by the solver within the specified time limit.

An alternative is to obtain an approximate solution through the Fix & Relax approach (Dillenberger *et al.* 1992). This involves the solution of a series of partially relaxed MIPs, each with a number of 0/1 variables that is small enough to be quickly and optimally solved by conventional branch-and-bound methods. As the series progresses, each set of 0/1 variables is permanently fixed at their solution values, and the relaxed variables are reduced in number, eventually disappearing. The procedure is broadly similar to a depth-first identification of an initial integer solution for a MIP model in a branch-and-bound search. Its big advantage is its speed. For model C, the formal procedure is as follows.

*Step* 1. Solve the partial linear programming relaxation, where the values of $\{y^1_{j_{0m}im1} \quad \forall i,m\}$ are constrained to be 0 or 1, while the other $y^n_{jim1}$ may vary continuously between 0 and 1.

*Step* 2. For $n = 2,\dots,N$, solve the partial linear programming relaxation, with the $\{y^1_{j_{0m}im1} \quad \forall i,m\}$ fixed at their 0 or 1 solution values from step 1, with the values of $\{y^2_{ikm1} \quad \forall i,m\}$ to $\{y^{n-1}_{ikm1} \quad \forall i,m\}$ fixed at their 0 or 1 solution values from previous applications of step 2, and with the values of $\{y^n_{ikm1} \quad \forall i,m\}$ constrained to be 0 or 1, while the remaining $y^{n+1}_{jim1}$ to $y^N_{jim1}$ may vary continuously between 0 and 1.

*Step* 3. For $t = 2,\dots,\tau$, repeat steps 1 and 2 for $y^1_{jimt}$ to $y^N_{jimt}$, fixing their values at those of the solutions in steps 1 and 2.

Note that each cycle of steps 1 and 2 involves solving $N$ problems with just $PM$ 0/1 variables each, as $PM(P-1)$ $y$-variables in each problem will newly have value 0 due to constraints (12).

Thus, the application of the cyclic Fix-&-Relax approach involves the solution of $N\tau$ MIPs with $PM$ 0/1 variables each. This is still a substantial number of 0/1 variables to solve in a MIP, but certainly much fewer than the $PM(PN\tau - P + 1)$ 0/1 variables involved in solving model C. For instance, scheduling up to three set-ups per period of 15 products on five machines rolling forward one period at a time involves solving model C with 2325 0/1 variables, while the Fix-&-Relax approach solves a succession of MIPs with only 75 0/1 variables each. As will be seen below, the Fix-&-Relax MIPs can be solved optimally and quickly for quite sizeable values of $P$ and $M$.

The benefits and limits of the Fix-and-Relax method were tested on model C, denoted method CFR, using the original demand as defined by expression (18). The results are shown in table 13 with the CPU times in minutes shaded in grey. To facilitate legibility, the solution values are expressed as a percentage of the value when just one set-up per period is allowed. Note that for the $N > 1$ method, CFR is, as we expect, able to obtain solutions much more rapidly than model C solved to optimality. For a given $N > 1$, where the optimal solution to C was found, the CFR solution is often worse (sometimes considerably so), occasionally very good, but always very much faster. As a result, method CFR is able to obtain solutions where model C cannot be solved to optimality in reasonable computing time.

However, observe in table 13 that method CFR itself begins to take too long to solve for problems with ten products on five machines. A possible response to this is to limit the amount of time spent in the branch-and-bound searches, namely, $10/N$ minutes per MIP or when the first feasible solution is found, whichever is the greater. Thus, we try to limit to ten minutes the CPU time spent not within each branch-and-bound search, but rather over all branch-and-bound searches at each rolling horizon, so that equal search time is expended at different values of $N$, making for fairer comparisons. If, for example, $N = 2$, then five minutes CPU time is allocated to each of the two branch-and-bound searches at each of the ten rolling horizons. If a

| Size | Method | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ | $N = 5$ |
|---|---|---|---|---|---|---|
| $M = 2$ | C | 100% | 0% | 0% | 0% | 0% |
| $P = 4$ | | *0.00* | *0.02* | *0.00* | *0.00* | *0.02* |
| | CFR | 100% | 10% | 0% | 0% | 0% |
| | | *0.00* | *0.00* | *0.00* | *0.00* | *0.00* |
| $M = 2$ | C | 100% | 28% | 1% | 0% | 0% |
| $P = 6$ | | *0.00* | *1.00* | *13.05* | *52.77* | *232.1* |
| | CFR | 100% | 47% | 51% | 44% | 36% |
| | | *0.00* | *0.02* | *0.00* | *0.03* | *0.03* |
| $M = 3$ | C | 100% | 1% | 0% | 0% | 0% |
| $P = 6$ | | *0.03* | *5.55* | *2.67* | *6.05* | *203.5* |
| | CFR | 100% | 28% | 11% | 1% | 7% |
| | | *0.03* | *0.00* | *0.03* | *0.02* | *0.12* |
| $M = 5$ | C | 100% | 0% | 0% | 0% | 0% |
| $P = 5$ | | *0.17* | *0.28* | *0.87* | *3.23* | *9.93* |
| | CFR | 100% | 0% | 5% | 0% | 0% |
| | | *0.17* | *0.02* | *0.02* | *0.03* | *0.18* |
| $M = 5$ | C | 100% | ? | ? | ? | ? |
| $P = 10$ | | *37.88* | *too long* | *too long* | *too long* | *too long* |
| | CFR | 100% | 25% | 0% | 0% | 0% |
| | | *37.88* | *12.58* | *1.32* | *0.97* | *1.70* |
| $M = 15$ | C | 100% | 66% | 57% | 35% | 27% |
| $P = 15$ | | *74.03* | *100.0* | *328.6* | *479.9* | *1123* |
| (*) | CFR | 100% | 65% | 34% | 22% | 8% |
| | | *74.02* | *42.32* | *28.55* | *29.20* | *12.33* |
| $M = 15$ | C | 100% | 77% | 61% | ? | ? |
| $P = 20$ | | *92.67* | *107.5* | *282.2* | *> 800* | *too long* |
| (*) | CFR | 100% | 66% | 45% | 36% | 24% |
| | | *92.67* | *52.17* | *40.97* | *45.95* | *57.65* |

Table 13. Methods C and CFR compared.

feasible solution is not found within the allocated time limit, then the search continues until one is identified.

Such results are shown by (*) for 15 and 20 products on five machines in table 13. Observe that, as more set-ups are permitted, i.e. as $N$ increases, the solution quality of model C improves, but unfortunately needs impracticable amounts of CPU time to find even a feasible solution. However, the solution quality of method CFR improves even faster than that of model C, and takes far less time. The 12.33 minutes taken to obtain the good CFR result with 15 products and five set-ups per period is the CPU time needed for all ten rolling horizon runs—the mean time per horizon is only a tenth of this, i.e. 1 minute and 14 seconds, a very practicable amount of time. The corresponding time for 20 products is over 5 minutes, not quite so practicable. However, bear in mind that the CPU time partially depends on the internal heuristics used to find an initial solution by a particular MIP solver, XPRESS-MP (1997) in

| $P$ | $N=1$ | $N=2$ | $N=3$ | $N=4$ | $N=5$ |
|---|---|---|---|---|---|
| 20 | 100% | 66% | 45% | 36% | 24% |
| | *92.67* | *52.17* | *40.97* | *45.95* | *57.65* |
| 30 | 100% | 62% | 42% | 39% | 30% |
| | *100.00* | *90.33* | *57.62* | *80.42* | *230.40* |
| 40 | 100% | 72% | 44% | 41% | 28% |
| | *100.00* | *67.43* | *99.10* | *350.57* | *221.80* |
| 50 | 100% | 79% | 87% | 91% | 82% |
| | *110.08* | *70.72* | *187.92* | *456.35* | *683.58* |

Table 14. Method CFR with five machines under limited CPU time.

this case. An alternative solver, such as ILOG CPLEX (1999), would give a different initial solution and CPU time as well as different optimal solution times, due to its own distinct implementation of the branch-and-bound approach. Indeed, initial tests suggest that the internal heuristic of CPLEX (version 6.5) is able to find an initial solution much more quickly than XPRESS-MP (release 10.07). Heeding such caveats, we now consider the performance of the CFR method under limited CPU time.

Table 14 shows snapshot tests for method CFR on larger sized problems (20, 30, 40 and 50 products on five machines). Note that for 20, 30 and 40 products, a greater number of permitted set-ups results in a better quality of solution, but that the CPU time necessary for XPRESS-MP to find just an initial feasible solution is becoming too large for practical purposes, well beyond the maximum we tried to impose. When 50 products are involved, the quality of the solution actually worsens for more than two permitted set-ups and the CPU time needed to find an initial solution is quite unacceptable. We could try using another solver such as CPLEX, relying on its internal heuristics to find a good initial solution within a reasonable time, but we would still be left with little more than initial feasible solutions rather than good ones. An alternative approach is considered in the concluding remarks below.

## 6. Conclusions

In this article we developed a mixed integer programming model for the multi-product lot-sizing problem with sequence-dependent set-up times. The formulation allows for multiple set-ups per planning period, a very flexible feature not considered in previous research. Multiple set-ups are needed when there are more products than machines and demand exists for all or most products in every period. It is a highly complex problem and difficult to solve to optimality for anything other than very small problems. However, as the backorder penalties are usually subjective estimates, and demand forecasts are invariably updated as planning horizons are rolled forward, an optimal solution to this exact model is not necessary. Consequently, we can replace the exact model by a smaller and faster approximate model that includes integer variables only for the planning periods up to the time of the next rolling horizon application of the model. Four varieties of alternative models (C to F) were tested and found to replicate the outcomes of the exact model quite well. Two of the alternative models in particular (C and D) were found to provide the best rolling horizon solutions. In particular, these models gave the best solutions with least computing time when the number of set-ups permitted per period was $\left(P/M+1\right)$.

However, there are limits to the size of problem that the approximate model can solve in practical time. A limit on the time spent searching for a particular solution can be imposed, an approach which is easy to implement in most MIP solvers, but the computational results above suggest that, for medium to large problems, impracticable amounts of time will be spent just identifying a feasible solution. An alternative approach now being explored by the authors is to use metaheuristic solution methods (Osman and Laporte 1996) to find good-quality solutions to the exact model A or to the approximate models C and D. The value in developing the approximate models is that, being of considerably smaller dimension, they need far less computing effort to identify near-optimal solutions than the exact model, be it through mathematical programming or metaheuristic approaches. The tests indicate that industrial users can obtain much faster responses with only a small loss of decision quality.

## References

AROSIO, M. and SIANESI, A., 1993, A heuristic algorithm for master production scheduling generation with finite capacity and sequence dependent set-ups. *International Journal of Production Research*, **31**, 531–553.

BAKER, K. R., 1977, An experimental study of the effectiveness of rolling schedules in production planning. *Decision Sciences*, **8**, 1927; 1979, An analytical framework for evaluating rolling horizon schedules. *Management Science*, 341–351; 1993, Requirements planning. In S. C. Graves (ed), *Handbooks in Operations Research and Management Science* (New York: Elsevier Science Publishers), **4**, pp. 571–627.

CLARK, A. R., 1998, Batch sequencing and sizing with regular varying demand. *Production Planning and Control*, **9**, 260–266.

DILLENBERGER, C., WOLLENSAK, A. and ZHANG, W., 1992, On practical resource allocation for production planning and scheduling with different set-up products. Draft paper, German Manufacturing Technology Centre, IBM, Sindelfingen, Germany.

DREXL, A. and HAASE, K., 1995, Proportional lotsizing and scheduling. *International Journal of Production Economics*, **40**, 73–87.

DREXL, A. and KIMMS, A., 1997, Lot sizing and scheduling—survey and extensions. *European Journal of Operational Research*, **99**, 221–235.

FOURER, R., GAY, D. M. and KERNIGHAN, B. W., 1993, *AMPL—a modeling language for mathematical programming* (Danvers, Massachusetts: Boyd and Fraser).

HEUTS, R. M. J., SEIDEL, H. P. and SELEN, W. J., 1992, A comparison of two lot sizing-sequencing heuristics for the process industry. *European Journal of Operational Research*, **59**, 413–424.

ILOG CPLEX 6.5 USER'S MANUAL, 1999, ILOG S.A., BP 85, 9 Rue de Berdun, 94253 Gentilly Cedex, France; http://www.ilog.com.

KADIPASAOGLU, S. N. and SRIDHARAN, S. V., 1997, Measurement of instability in multi-level MRP systems. *International Journal of Production Research*, **35**, 713–737.

MAES, J. and VAN WASSENHOVE, L. N., 1986, Multi item single level capacitated dynamic lotsizing heuristics: A computational comparison (Part II: Rolling horizon). *IIE Transactions*, **18**, 124–129; 1988, Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *Journal of the Operational Research Society*, **39**, 991–1004.

OSMAN, I. H. and LAPORTE, G., 1996, Metaheuristics: A bibliography. *Annals of Operations Research*, **63**, 513–623.

OVACIK, I. M. and UZSOY, R., 1995, Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent set-up time. *International Journal of Production Research*, **33**, 3173–3192.

SMITH-DANIELS, V. L. and SMITH-DANIELS, D. E., 1986, A mixed integer programming model for lot sizing and sequencing packaging lines in the process industries. *IIE Transactions*, **18**, 278–285.

TOKLU, B. and WILSON, J. M., 1995, An analysis of multi-level lot-sizing problems with a bottleneck under a rolling horizon schedule environment. *International Journal of Production Research*, **33**, 1835–1847.

WOLSEY, L. A., 1995, Progress with single-item lot-sizing. *European Journal of Operational Research*, **86**, 395–401; 1997, MIP modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research*, **99**, 154–165.

XPRESS-MP Workstation Model Builder Release 10.02 and Integer Optimiser Release 10.07, 1997, Dash Associates, Blisworth House, Blisworth, Northamptonshire NN7 3BX, UK.