An Efficient MIP Model for the Capacitated Lot-sizing and Scheduling Problem with Sequence-dependent Setups

András Kovács[a,c], Kenneth N. Brown[a,b], S. Armagan Tarim[b,d]

[a]Cork Constraint Computation Centre, University College Cork, Ireland

[b]Centre for Telecommunications Value-Chain Research, Ireland

[c]Computer and Automation Research Institute, Budapest, Hungary

[d]Hacettepe University, Department of Management, Ankara, Turkey

E-mail addresses: akovacs@sztaki.hu, k.brown@cs.ucc.ie, armagan.tarim@hacettepe.edu.tr

September 15, 2008

Abstract

This paper presents a novel mathematical programming approach to the single-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times and setup costs. The approach is partly based on the earlier work of Haase and Kimms (2000) which determines during pre-processing all item sequences that can appear in given time periods in optimal solutions. We introduce a new mixed-integer programming model in which binary variables indicate whether individual items are produced in a period, and parameters for this program are generated by a heuristic procedure in order to establish a tight formulation. Our model allows us to solve in reasonable time instances where the product of the number of items and number of time periods is at most 60–70. Compared to known optimal solution methods, it solves significantly larger problems, often with orders of magnitude speedup.

Keywords: Lot-sizing, scheduling, sequence-dependent setups, mixed-integer programming.

1 Introduction

This paper considers the lot-sizing and scheduling problem involving production of multiple items on a single finite capacity machine with sequence-dependent setup costs and setup times. In this problem, the decision maker must decide which items to produce in which periods, and must specify the exact production sequence and production quantities to satisfy deterministic dynamic demand over multiple periods that span a planning horizon, in order to minimise the sum of setup and inventory holding costs. The consideration of capacity limitations, significant sequence-dependent setup costs and non-zero setup times exacerbates the inherent difficulty in solving lot-sizing and scheduling problems and restricts the problem size that can be tackled in reasonable time. Ignoring these features when planning production aggravates costs and reduces productivity, particularly in process industries such as chemicals, drugs and pharmaceuticals, pulp and paper, food and beverage, textiles, or ceramics. Other examples include discrete manufacturing in industries such as aerospace, defense and automotive. All such manufacturers could benefit significantly from progress in this research area.

Recent work by Haase and Kimms (2000) proposes an exact optimisation approach to the problem. Their approach is based upon a mixed-integer programming (MIP) formulation. They start by generating all possible efficient sequences of items, and then use binary variables in the MIP to denote whether a sequence is selected for a given time period. However, the applicability of their approach is limited to either a small number of items or a short planning horizon. In this paper, we present an alternative model, which also uses pre-generated efficient sequences, but employs binary variables to indicate whether or not an item is produced in a given period. This yields smaller models, but makes it harder to express constraints on the setup costs. A naive formulation of these constraints gives loose LP relaxations, and hence an inefficient model. We then develop a heuristic algorithm which generates much tighter constraints.

We show experimentally that the proposed MIP model outperforms all previously known optimisation approaches to the capacitated lot-sizing and scheduling problem (CLSP) with sequence-dependent setups. It gives up to two orders of magnitude speedup in solution time over the Haase and Kimms model, and can solve larger instances. We also show that the efficient sequences can be generated more effectively, and that the same underlying model can be applied to a number of variants of the problem with similar time performance. The practical implications of these are significant.

The paper is organised as follows. In Section 2 we define the CLSP with sequence-dependent setups. Section 3 summarises previous work on this problem. In Section 4, we present an efficient dynamic program (DP) for generating the set of item sequences that might be applied in time periods in optimal solutions. Afterwards, we define a new MIP formulation of the problem (Section 5), and evaluate its performance on a set of randomly generated problem instances (Section 6). Finally, conclusions are drawn and directions of future research are outlined.

2 Problem definition

The *capacitated lot-sizing and scheduling problem with sequence-dependent setup times and costs* (CLSPSD) involves N_I different *items* able to be manufactured on a single machine over a series of N_T *time periods*. In each time period t, we must decide how many units x_t^i of each item i to produce. Since we have a single machine available, the production of different items within a time period must be sequenced. However, switching from item i to item j requires a *setup*, which occupies $T^{i,j}$ units of the capacity in the given time period, and incurs $C^{i,j}$ cost. Producing one lot of item i employs the machine for $p^i x_t^i$ time, which is thus proportional to the lot size. The sum of all setup and production times within a time period cannot exceed the available *capacity* C_t in that period. The *demand* d_t^i for each item and time period is fully known in advance, and must be met exactly, either from production in that period or from excess produced in previous periods. The cost of a solution is composed of the sequence-dependent setup costs and the *inventory holding costs* h^i per excess unit of item i at the end of every time period.

The objective is to choose the production quantities and production sequences for each time period to meet the demand while minimising the total cost. The following assumptions are made.

(i) The cost of switching from item i to item j can be computed as $C^{i,j} = q^i + r\,T^{i,j}$, where q^i is the *direct setup cost* of switching to item i, and r is the *time-proportional setup coefficient*.

(ii) Setup times satisfy the triangle inequality, i.e., $T^{i,j} \leq T^{i,k} + T^{k,j}$. Due to the previous assumption, the triangle inequality holds also for the setup costs.

(iii) The setup states are carried over from one time period to the next. It is allowed to switch from one item to another in idle periods (i.e., when no production occurs), but it incurs the same setup cost as if the item was produced.

(iv) Setups are performed within one time period. This also implies that a problem instance is feasible only if $T^{i,j} \leq C_t$ holds for all relevant pairs of items i and j and time period t.

In the micro-level representation of the solutions of CLSPSD, several items can be produced in each time period on the same machine, sequentially one after the other. Note that since setup times and costs are sequence-dependent, the sequence of item production in a period affects both feasibility and cost, and is a crucial issue for generating optimal solutions. Choosing a sequence of items $\sigma = (i_{k_1}, i_{k_2}, ..., i_{k_n})$ for production in time period t means that the machine is set up to produce item $\sigma[1] = i_{k_1}$ at the beginning of t; after producing a certain amount of $\sigma[1]$, a changeover from $\sigma[1]$ to $\sigma[2]$ occurs, and this continues until the end of time period t. At that point, the machine will be set up to produce item $\sigma[n_\sigma] = i_{k_n}$, where n_σ denotes the number items in sequence σ. Since setup states are carried over, the sequence applied in time period $t+1$ has to begin with item $\sigma[n_\sigma]$.

Note that applying sequence σ in t does not imply that a positive amount of items $\sigma[1]$ or $\sigma[n_\sigma]$ are actually produced in period t. It might happen that item $\sigma[1]$ was produced in period $t-1$, but switching from $\sigma[1]$ to $\sigma[2]$ takes place in t, or analogously, the machine is set up to item $\sigma[n_\sigma]$ so that period $t+1$ can start immediately with production. Hence, for the sake of simplicity, when saying an item is *produced* in a time period, we allow the production of zero quantities as well. At the same time, since the triangle inequality holds for the setup times, producing an empty lot of item $\sigma[k]$ for $k = 2, ..., n-1$ would lead to sub-optimality.

The micro-structure of a time period is illustrated in Fig. 1. Observe that the overall capacity required in time period t can be divided into two components. First, the capacity spent for setups, the amount of which depends only on the sequence applied, but

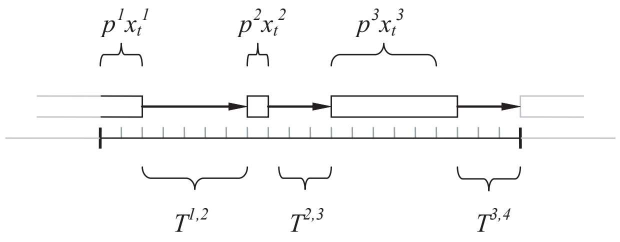

Figure 1: A possible micro-structure of period t if sequence (i_1, i_2, i_3, i_4) is applied.

not on the actual lot sizes. In contrast, the capacity required for production is proportional to the amounts of each item produced. The sum of these two components must not exceed the capacity available in the given time period.

3 Previous work on CLSPSD

Capacitated lot-sizing problems and their different variants are widely studied in the literature of operations research. A review of various lot-sizing and scheduling models, including small-bucket, large-bucket, and continuous time formulations is presented in (Drexl and Kimms, 1997). Another survey by Belvaux and Wolsey (2001) discusses modelling options for setups and other practical requirements in small-bucket as well as large-bucket representations. The authors also propose valid inequalities for the efficient MIP formulation of these problem variants. Karimi et al. (2003) present a classification scheme for capacitated lot-sizing problems and give references to different exact and heuristic solution approaches. The strong NP-hardness of the multi-item capacitated lot-sizing and scheduling problem (CLSP) has been proven by Chen and Thizy (1990).

The literature of CLSP with sequence-dependent setup times and setup costs is considerably scarcer. A local search approach combined with dual re-optimisation for CLSPSD has been proposed by Meyr (2000). This approach was extended to the case of parallel machines by the same author in (Meyr, 2002). Exact optimisation approaches to CLSPSD have been suggested by Gupta and Magnusson (2005), and Haase and Kimms (2000), both using MIP techniques. Gupta and Magnusson define a MIP that makes not only the lot-sizing, but also all sequencing decisions within time periods on the fly. Although this approach makes it possible to relax assumption (i) about the relation of setup times and costs, it allows us to find optimal solutions for very small instances only, e.g., with 3 items and 3 time periods. For larger problems, a heuristic procedure is proposed in the same paper.

Below, we discuss in detail the approach proposed by Haase and Kimms (2000), but use a slightly different terminology. The authors analysed the item sequences that can occur in given time periods in an optimal solution of a CLSPSD, and introduced a MIP to choose one such sequence for each time period. They pointed out that item sequences can be classified into *scenarios*. A scenario $\alpha = \langle i_f, i_l, I \rangle$ is identified by the first and last items i_f and i_l, and the set I of all items produced in the time period. Now, for each scenario α, there exists a sequencing σ_α of the item set I with $\sigma_\alpha[1] = i_f$, $\sigma_\alpha[n_{\sigma_\alpha}] = i_l$ that minimises the setup time incurred by the sequence, i.e.,

$$T_{\sigma_\alpha} = \sum_{k=1}^{n_{\sigma_\alpha}-1} T^{\sigma_\alpha[k],\sigma_\alpha[k+1]}.$$

This sequence σ_α is called the *efficient sequence* corresponding to scenario α. Note that it follows from assumption (i) that the same σ_α also minimises the setup cost. The authors have shown that the application of a sequence that is not efficient for the given scenario in a solution leads to sub-optimality. Consequently, the set of all item sequences that can be applied in optimal solutions consists of the above defined efficient sequences. If there are several efficient sequences for a scenario, then one can be chosen arbitrarily. The number of scenarios – and therefore, of efficient sequences as well – is

$$N_S = N_I(N_I - 1)\, 2^{N_I-2} \;+\; N_I\, 2^{N_I-1},$$

where the first and second parts stand for the number of scenarios with $i \not\equiv j$ and $i \equiv j$, respectively. Note that N_S grows exponentially with N_I. Unless some special inference can be performed on the demands or costs, any efficient sequence can take part in an optimal solution: consider an instance with two time periods, an initial setup state of i_f, and demands such that $d_1^i > 0$ iff $i \in I$ and $d_2^{i_l} = C_2/p^{i_l}$. Now, it is easy to see that if holding costs are high, then scenario $\langle i_f, i_l, I\rangle$, and its corresponding efficient sequence must be applied in the first time period in the optimal solution of the instance. The same example illustrates that scenarios with $i_l \equiv i_f$ might appear in optimal solutions[1]. Idle periods can be represented by a scenario of the form $\langle i, i, \{i\}\rangle$ with zero production.

Haase and Kimms (2000) argue that each efficient sequence can be determined by solving a *travelling salesman problem* (TSP) corresponding to the scenario, where the setup time of the sequence equals the value of the optimal TSP solution. Although solving N_S separate TSP instances can be time consuming, in an industrial application this pre-processing step has to be carried out only when the products of the factory or the setup times change.

Having generated all efficient sequences, Haase and Kimms (2000) define a MIP containing binary variables v_t^σ to indicate whether sequence σ is applied for production in time period t. Consequently, we call this MIP a *sequence-related* formulation. Other variables, x_t^i standing for the amount of item i produced in period t and s_t^i denoting the stock of item i at the end of period t, and the constraints are as in the classical MIP representation of CLSP (Drexl and Kimms, 1997). Hence, this MIP can be described using $O(N_S N_T)$ binary variables, $O(N_I N_T)$ real variables, and $O(N_I N_T)$ inequalities. The authors report that this MIP is capable of solving instances with up to 3 items and 15 time periods, or 10 items and 3 periods.

[1]Apparently, many papers in the sequence-dependent lot-sizing literature ignore this phenomenon, see e.g., (Gupta and Magnusson, 2005) and (Haase and Kimms, 2000).

4 A DP for computing efficient sequences

In this section, we show that the set of all efficient sequences can be generated by a quick DP, without solving a separate TSP for each of the N_S scenarios. A similar algorithm was originally proposed by Bellman (1962) for solving individual TSP instances.

The DP is based on the observation that if $\sigma = (i_1, i_2, ..., i_{n-1}, i_n)$ is an efficient sequence for scenario $\alpha = \langle i_1, i_n, \{i_1, ..., i_n\}\rangle$, then the shorter sequence $\sigma' = (i_1, i_2, ..., i_{n-1})$ is an efficient sequence for the scenario $\alpha' = \langle i_1, i_{n-1}, \{i_1, ..., i_{n-1}\}\rangle$. From this, it follows that the efficient sequence for scenario α can be constructed from one of the efficient sequences for scenarios $\langle i_1, i_k, \{i_1, ..., i_{n-1}\}\rangle,\ i_k \in \{i_2, ..., i_{n-1}\}$, by appending item i_n.

The DP presented in Fig. 2 exploits this consequence. It constructs optimal TSP solutions for all the scenarios in increasing order of the number of items contained. For each non-trivial scenario α, it computes Θ, the set of candidate sequences that – by appending one item – can become an efficient sequence for α, and chooses the one with minimal setup time. The time complexity of the algorithm is $O(N_S N_I)$.

```
PROCEDURE ComputeEfficientSequences()
    FORALL scenario $\alpha = \langle i_f, i_l, I\rangle$ ordered by increasing $|I|$
      IF $|I| \leq 2$ THEN
        $\sigma_\alpha :=$ the only sequence that realises $\alpha$
      ELSE
        IF $i_f \equiv i_l$ THEN
          $\Theta :=$ efficient sequences for scenarios
               $\{\langle i_f, i'_l, I\rangle \mid i'_l \in I \setminus \{i_f\}\}$
        ELSE
          $\Theta :=$ efficient sequences for scenarios
               $\{\langle i_f, i'_l, I \setminus \{i_l\}\rangle \mid i'_l \in I \setminus \{i_f, i_l\}\}$
        $\sigma_\alpha := \sigma + i_l$, where $\sigma \in \Theta$ is the the sequence that
          minimises $T_\sigma + T^{\sigma[n_\sigma], i_l}$
```

Figure 2: A dynamic program for computing efficient sequences.

5 An efficient MIP model for CLSPSD

The drawback of the sequence-related MIP representation of CLSPSD presented in Section 3 is that the number of binary variables grows exponentially with the number of items, which leads to poor scaling. Below we define an *item-related* representation with only $O(N_I N_T)$ binary variables, y_t^i deciding if item i is produced in period t, and z_t^i indicating if item i is produced *last* in period t. Note that these variables unambiguously identify the sequence applied in each time period t: it is the efficient sequence corresponding to scenario $\langle i_f, i_l, I\rangle$, where $I = \{i \mid y_t^i = 1\}$, i_f is the unique item with $z_{t-1}^{i_f} = 1$ and i_l with $z_t^{i_l} = 1$.

While most constraints of CLSPSD can be expressed using variables and inequalities of the classical MIP model of lot-sizing (Drexl and Kimms, 1997) (see also Sect. 5.2), the capacity constraint and the objective function requires a different treatment: the sequence-dependent setup times and costs have to be considered in them. For this purpose, we introduce real variables u_t to denote the setup time incurred in time period t. The setup cost that occurs in period t can be expressed from u_t using the linear expression introduced in assumption (i).

Now, the setup time u_t has to be related to the sequence applied in period t. Since sequences are not explicitly modelled in the MIP, this can be done by means of linear inequalities on variables y_t^i, z_{t-1}^i, and z_t^i. They will take the form

$$u_t \geq \bar{L}\bar{y}_t + \bar{M}\bar{z}_{t-1} + \bar{N}\bar{z}_t + K,$$

where $\bar{L}$, $\bar{M}$, $\bar{N}$, and K are constants to be defined later. These inequalities provide lower bounds on u_t. The set of inequalities is sound if, no matter which efficient sequence σ is chosen for a period t, the strongest lower bound among *all these inequalities* on u_t is exactly T_σ.

We will define one such inequality for each sequence σ, and call it the σ-inequality. The value of its r.h.s. – with the substitution of the variables according to an arbitrary sequence σ' – is the σ'*-substitution* of this inequality. Finally, if the σ'-substitution of a σ-inequality is smaller than, equal to, or larger than $T_{\sigma'}$, then we call this inequality *non-constraining*, *constraining*, or *over-constraining* on σ', respectively.

Now, a sufficient condition for the soundness of the set of σ-inequalities can be stated as follows. Firstly, for each sequence σ, the σ-inequality has to be constraining on σ. Secondly, all other inequalities must not be over-constraining on σ. The standard mathematical programming method for specifying such a set of inequalities is the use of so-called big-M constraints (Williams, 1999). The big-M formulation of the σ-inequality for a given σ takes the form

$$\begin{aligned} u_t \geq\ & T_\sigma - \sum_{i\in\sigma}(1-y_t^i)L_\sigma^i & &+ \sum_{i\notin\sigma} y_t^i L_\sigma^i \\ & - (1 - z_{t-1}^{\sigma[1]})M_\sigma^{\sigma[1]} & &+ \sum_{i\neq\sigma[1]} z_{t-1}^i L_\sigma^i \\ & - (1 - z_t^{\sigma[n_\sigma]})N_\sigma^{\sigma[n_\sigma]} & &+ \sum_{i\neq\sigma[n_\sigma]} z_t^i M_\sigma^i, \end{aligned}$$

where L_σ^i, M_σ^i, and N_σ^i are sufficiently large coefficients. This can be rewritten as

$$u_t \geq T_\sigma + \sum_i (y_t^i L_\sigma^i + z_{t-1}^i M_\sigma^i + z_t^i N_\sigma^i) - (\sum_{i\in\sigma} L_\sigma^i + M_\sigma^{\sigma[1]} + N_\sigma^{\sigma[n_\sigma]}).$$

It is easy to see from the original form of the inequality that if variables are substituted according to sequence σ, then all addends on the r.h.s. except for T_σ are zero, for an arbitrary choice of coefficients L_σ^i, M_σ^i, and N_σ^i. Therefore, the σ-inequality is constraining on σ. Also note that the coefficients can be selected in a way that the inequality is not over-constraining for any other sequence, e.g., with

$$L^i_\sigma = \begin{cases} \infty & \text{if } i \in \sigma \\ 0 & \text{otherwise} \end{cases} \quad M^i_\sigma = \begin{cases} \infty & \text{if } i = \sigma[1] \\ 0 & \text{otherwise} \end{cases} \quad N^i_\sigma = \begin{cases} \infty & \text{if } i = \sigma[n_\sigma] \\ 0 & \text{otherwise.} \end{cases}$$

However, this choice of coefficients would lead to extremely weak LP relaxations. Next, we will show how to generate coefficients for tighter relaxations, and hence more efficient MIPs.

5.1 Computing coefficients for the setup time inequalities

Note that the challenge of computing appropriate coefficients for the setup time inequalities is analogous to the problem of *lifting* (Marchand et al., 2002): coefficients are sought for inequalities of known form so as to gain tight LP relaxations. Hence, we take an approach similar to that used in sequential lifting. We assign initial values to L^i_σ, M^i_σ, and N^i_σ, and then set the coefficients one by one to their extreme value permitted by the non-over-constraining condition. We begin by defining the following sets of sequence pairs:

$$\begin{aligned}
\mathcal{L}^i \;&:= \{\langle\sigma,\sigma'\rangle \mid i \in \sigma \ \wedge\ i \notin \sigma' \ \wedge\ \forall j \neq i : (j \in \sigma \Leftrightarrow j \in \sigma') \\
&\quad\ \wedge\ \sigma[1] = \sigma'[1] \ \wedge\ \sigma[n_\sigma] = \sigma'[n_{\sigma'}]\} \\
\mathcal{M}^{i,j} &:= \{\langle\sigma,\sigma'\rangle \mid \forall k : (k \in \sigma \Leftrightarrow k \in \sigma') \\
&\quad\ \wedge\ \sigma[1] = i \ \wedge\ \sigma'[1] = j \ \wedge\ \sigma[n_\sigma] = \sigma[n_\sigma]\} \\
\mathcal{N}^{i,j} &:= \{\langle\sigma,\sigma'\rangle \mid \forall k : (k \in \sigma \Leftrightarrow k \in \sigma') \\
&\quad\ \wedge\ \sigma[1] = \sigma'[1] \ \wedge\ \sigma[n_\sigma] = i \ \wedge\ \sigma'[n_{\sigma'}] = j\}
\end{aligned}$$

Broadly speaking, $\mathcal{L}^i$ is the set of all pairs of efficient sequences σ and σ' that only differ in that item i is a member of σ, but not of σ'. Similarly, members of $\mathcal{M}^{i,j}$ differ only in their first item, while those of $\mathcal{N}^{i,j}$ in their last item. Then, let L^i_{max} be the largest difference between the setup times of the two members of a sequence pair in $\mathcal{L}^i$, and similarly:

$$L^i_{max} := \max_{\langle\sigma,\sigma'\rangle \in \mathcal{L}^i} T_\sigma - T_{\sigma'}$$

$$L^i_{min} := \min_{\langle\sigma,\sigma'\rangle \in \mathcal{L}^i} T_\sigma - T_{\sigma'}$$

$$M^{i,j}_{min} := \min_{\langle\sigma,\sigma'\rangle \in \mathcal{M}^{i,j}} T_\sigma - T_{\sigma'}$$

$$N^{i,j}_{min} := \min_{\langle\sigma,\sigma'\rangle \in \mathcal{N}^{i,j}} T_\sigma - T_{\sigma'}$$

Now, the following inequality holds for each pair of sequences σ and σ':

$$T_{\sigma'} \ \geq\ T_\sigma \ -\ \sum_{i \in \sigma \wedge i \notin \sigma'} L^i_{max} \ +\ \sum_{i \notin \sigma \wedge i \in \sigma'} L^i_{min} \ +\ M^{\sigma'[1],\sigma[1]}_{min} \ +\ L^{\sigma'[n_{\sigma'}],\sigma[n_\sigma]}_{min}$$

By introducing variables y^i_t and z^i_t to characterise sequence σ', we receive that the following inequality holds independently of the sequence σ' applied in time period t:

$$u_t \ \geq \ T_\sigma - \sum_{i\in\sigma}(1-y_t^i)L_{max}^i + \sum_{i\notin\sigma} y_t^i L_{min}^i + \sum_{i\neq\sigma[1]} z_{t-1}^i M_{min}^{i,\sigma[1]} + \sum_{i\neq\sigma[n_\sigma]} z_t^i L_{min}^{i,\sigma[n_\sigma]}$$

Therefore, by choosing the coefficients of the σ-inequality in the following way, the inequality will not be over-constraining on any sequences:

$$L_\sigma^i = \begin{cases} L_{max}^i & \text{if } i \in \sigma \\ L_{min}^i & \text{otherwise} \end{cases}$$

$$M_\sigma^i = \begin{cases} 0 & \text{if } i = \sigma[1] \\ M_{min}^{i,\sigma[1]} & \text{otherwise} \end{cases}$$

$$N_\sigma^i = \begin{cases} 0 & \text{if } i = \sigma[n_\sigma] \\ N_{min}^{i,\sigma[n_\sigma]} & \text{otherwise} \end{cases}$$

```
GLOBAL SET $\Omega := \emptyset$

PROCEDURE TightenAllCoeffs()
    FORALL sequence $\sigma$
      IF $\sigma \notin \Omega$ THEN
        TightenCoeffs($\sigma$)

PROCEDURE TightenCoeffs(sequence $\sigma$)
    FORALL sequence $\sigma'$
      $v[\sigma'] :=$ the $\sigma'$-substitution of the $\sigma$-inequality
    FORALL item $i$
      $\Theta := \{\sigma' \mid (i \in \sigma) \neq (i \in \sigma')\}$
      FORALL $\sigma' \in \Theta$
        $d_{\sigma'} := T_{\sigma'} - v[\sigma']$
        IF $d_{\sigma'} = 0$ THEN
          $\Omega := \Omega \cup \{\sigma'\}$
      $d := \min_{\sigma' \in \Theta} d_\sigma$
      IF $d > 0$ THEN
        IF $i \in \sigma$ THEN
          $L_\sigma^i := L_\sigma^i - d$
        ELSE
          $L_\sigma^i := L_\sigma^i + d$
        FORALL $\sigma' \in \Theta$
          $v[\sigma'] := v[\sigma'] + d$
```

Figure 3: A heuristic algorithm for tightening the coefficients.

A further tightening of the LP relaxations requires decreasing the coefficients L_σ^i where $i \in \sigma$, $M_\sigma^{\sigma[1]}$, and $N_\sigma^{\sigma[n_\sigma]}$, and increasing coefficients L_σ^i where $i \notin \sigma$, M_σ^i where $i \neq \sigma[1]$, and N_σ^i where $i \neq \sigma[n_\sigma]$. Note that while the coefficients in one

σ-inequality are interconnected by the non-over-constraining condition, coefficients belonging to different sequences can be considered independently.

In Fig. 3, we sketch a procedure that follows the above scheme, and by iterating over sequences σ and items i, computes the extreme values of the coefficients L^i_σ allowed by the non-over-constraining condition. Note that this procedure can turn a given σ-inequality into one constraining for several sequences $\sigma' \neq \sigma$ as well. Hence, for these sequences σ', the σ'-inequality becomes redundant, and can be omitted from the MIP. Although this might result in slightly looser LP relaxations, it leads to lower solution times by decreasing the size of the MIP. Such sequences σ' are therefore added to the set Ω, and ignored during the tightening procedure as well. The algorithm can be implemented to run in $O(N_S^2)$ time. Coefficients M^i_σ and N^i_σ can be tightened in a similar way.

5.2 The mixed-integer program

After all the above considerations, we are ready to present our MIP model for CLSPSD:

For *parameters*

d^i_t :	the demand for item i at the end of time period t
C_t :	the capacity available in time period t
h^i :	the holding cost for one unit of item i, from one period to the next
p^i :	the capacity required to produce one unit of item i
q^i :	the direct cost of setting up the machine for item i
r :	the time-proportional setup cost coefficient
T_σ :	the total setup time incurred by sequence σ
$L^i_\sigma,\ M^i_\sigma,\ N^i_\sigma$:	setup coefficients (see Section 5.1)

and *decision variables*

x^i_t :	the amount of item i produced in period t; $x^i_t \geq 0$
y^i_t :	the *produced* variable, $y^i_t = 1$ if item i is produced in period t; $y^i_t \in \{0,1\}$
z^i_t :	the *produced-last* variable, $z^i_t = 1$ if item i is produced last in period t (and also first in period $t+1$); $z^i_t \in \{0,1\}$
w^i_t :	the *setup* variable, $w^i_t = 1$ if a setup to item i occurs in period t; $w^i_t \geq 0$, its integrality is implied
s^i_t :	the stock of item i held at the end of period t; $s^i_t \geq 0$
u_t :	the total setup time incurred in time period t; $u_t \geq 0$

Minimise

$$\sum_{t,i} h^i s^i_t \ + \ \sum_t r u_t \ + \ \sum_{t,i} q^i w^i_t \tag{1}$$

subject to

$$\forall i,t \qquad s^i_{t-1} + x^i_t - d^i_t = s^i_t \tag{2}$$

$$\forall i,t \qquad C_t y^i_t \geq p^i x^i_t \tag{3}$$

$$\forall t \qquad u_t + \sum_i p^i x^i_t \leq C_t \tag{4}$$

$$\forall t \qquad \sum_i z^i_t = 1 \tag{5}$$

$$\forall i,t \qquad z^i_t \leq y^i_t \tag{6}$$

$$\forall i,t \qquad z^i_{t-1} \leq y^i_t \tag{7}$$

$$\forall i,t \qquad w^i_t \geq y^i_t - z^i_{t-1} \tag{8}$$

$$\forall i, i' \neq i, t \qquad w^i_t \geq z^i_t + y^{i'}_t - 1 \tag{9}$$

$$\begin{aligned}\forall t, \sigma \notin \Omega \qquad u_t \geq T_\sigma + \sum_i (y^i_t L^i_\sigma + z^i_{t-1} M^i_\sigma + z^i_t N^i_\sigma) - \\ - (\sum_{i\in\sigma} L^i_\sigma + M^{\sigma[1]}_\sigma + N^{\sigma[n_\sigma]}_\sigma)\end{aligned} \tag{10}$$

$$\forall i,t \qquad s^i_{t-1} \geq d^i_t (1 - y^i_t) \tag{11}$$

$$\forall i, i', t \qquad u_t \geq (y^i_t + y^{i'}_t - 1)\ \min(T^{i,i'}, T^{i',i}) \tag{12}$$

The objective (1) is to minimise the sum of the holding cost and the direct and time-proportional setup costs. Equality (2) ensures inventory balance, where s^i_0 specify the initial inventory levels. Inequality (3) states that an item can be produced only if the machine is set up for it. Inequality (4) describes the capacity constraint. Constraint (5) ensures that the machine is set up for exactly one item at the ends of time periods; the initial setup state is denoted by z_0.

Inequalities (6-9) describe the logical relations between variables y, z, and w. Namely, (6) and (7) state that if item i is produced first (z^i_{t-1}) or last (z^i_t), then it is produced (y^i_t). Inequality (8) ensures that if item i is produced (y^i_t), but not first (z^i_{t-1}), then a setup is performed (w^i_t). (9) states that a setup is required also if item i is produced last (z^i_t), but other items are produced, too. The setup time constraints (10) relate the setup time u_t to the scenario applied in time period t, as explained in the previous section. Note that only non-redundant setup time inequalities, i.e., those for $\sigma \notin \Omega$ have to be added to the MIP.

Lines (11) and (12) are valid inequalities. Namely, inequality (11) states that if item i is not produced in time period t, then the demand at the end of this period has to be satisfied from stock (see Belvaux and Wolsey, 2001). Constraint (12) gives a lower bound on the setup time incurred if at least two items are produced within the same time period. All in all, our MIP uses $2N_I N_T$ binary and $3N_I N_T + N_T$ real decision variables, and $O(N_S N_T)$ constraints.

5.3 Variants of the problem

The proposed approach to modelling sequence-dependent setups is applicable to many different variants of the CLSP addressed in the literature. Below, we discuss in detail the presence of backlogs and the zero-switch property.

Similarly to the case of the classical CLSP model without setup times (Belvaux and Wolsey, 2001), modelling backlogs requires the introduction of real decision variables $b_t^i \geq 0$ to denote the backlog of item i at the end of time period t, and parameters g^i for the backlogging cost of item i. Furthermore, the original objective function (1) has to be modified to (1a), the inventory balance constraint (2) to (2a), and valid inequality (11) has to be replaced by (11a):

$$\sum_{t,i} h^i s_t^i \;+\; \sum_t r u_t \;+\; \sum_{t,i} q^i w_t^i \;+\; \sum_{t,i} g^i b_t^i \tag{1a}$$

$$\forall i,t \quad s_{t-1}^i - b_{t-1}^i + x_t^i - d_t^i = s_t^i - b_t^i \tag{2a}$$

$$\forall i,t \quad s_{t-1}^i + b_t^i \geq d_t^i (1 - y_t^i) \tag{11a}$$

The problem variant where solutions must satisfy the zero-switch property was considered in order to enable a fair comparison to the MIP proposed by Haase and Kimms (2000). This property states that a new lot of a given item can only be scheduled when the inventory of that item is empty. This can be expressed by constraint (13), where Z is a big number, e.g., $Z = \max_i \sum_t d_t^i$. While the zero-switch property holds for optimal solutions of many uncapacitated lot-sizing problems, it can obviously lead to sub-optimality in capacitated problems like the CLSPSD:

$$\forall i, t \geq 2 \qquad s_{t-1}^i \leq Z(1 - y_t^i + z_{t-1}^i) \tag{13}$$

6 Experimental results

We ran experiments on a set of randomly generated problem instances in order to compare the performance of the MIP presented above to the best previously published optimisation approach. Experimental results achieved on several variants of the problem, i.e., with and without the zero-switch property, and with backlogging allowed are also presented below.

6.1 Comparison to (Haase and Kimms, 2000)

In order to provide a fair basis for the comparison to the MIP proposed by Haase and Kimms (2000), we generated problems in a similar fashion, used the same assumptions, and looked for solutions that satisfy the zero-switch property (see inequality (13)). However, we allowed sequences with identical first and last items, since otherwise ignoring this possibility as in (Haase and Kimms, 2000) may produce sub-optimal solutions.

A total of 1296 problem instances were generated systematically, by varying the number of items N_I between 3 and 10, choosing the number of time periods N_T from $\{4, 6, 8, ..., 20\}$, the time-proportional setup cost coefficient R from $\{50, 100, 200, 300, 400, 500\}$, and the capacity utilisation $U = \sum_t C_t / \sum_{t,i} p^i d_t^i$ from $\{0.4, 0.6, 0.8\}$. For each combination of the above parameters, one instance was created by choosing the demand d_t^i from $[0, 100]$, the holding cost h^i from $[2, 10]$, and the direct setup cost Q^i from $[100, 500]$ with uniform random distribution. Initial inventories were empty. Without loss of generality, the resource requirements p^i and also the initial setup state could be set to 1.

In order to obtain setup times that satisfy the triangle inequality, we generated for each item a point in the cube $[0, 10]^3$ with uniform distribution, and chose $T^{i,j}$ to be the rounded distance of the two points corresponding to items i and j. Finally, capacities C_t were set according to $C_t = \sum_i d_t^i / U$. Note that this formula ensures that a given portion determined by U of the overall capacity has to be spent on production, while it ignores setup times. Hence, the feasibility of the problem instances could not be guaranteed: one of the 1296 instances turned out to be infeasible, and was excluded from further experiments.

The algorithms for the two steps of the pre-processing (generating the sequences and determining the coefficients) were implemented in C++. The MIPs were encoded in CPLEX 10.0, and solved using its default solution strategy on a 2.0GHz Pentium IV computer with 1GB of RAM. A time limit of two hours was imposed for each MIP. Since pre-processing took less than 4 seconds even for the 10 items problem instances (and less than 1 second for smaller instances), pre-processing time was omitted. In order to save running time, we excluded instances with a given combination of N_I and N_T if none of the instances with smaller N_I and N_T could be solved by the same MIP.

The results are presented in Tables 1 and 2. Each row of the tables contains accumulated results for the 18 instances with the same number of items (N_I) and number of time periods (N_T). The second group of columns, under the heading *Solved (%)*, contains the percentage of instances that could be solved to optimality within the allotted time using the two MIPs. IR stands for the MIP proposed above using *item-related* binary variables, while SR for the MIP of Haase and Kimms (2000) using *sequence-related* variables. Finally, average solution times in seconds follow on the instances solved by IR and SR, respectively. Column IR* contains in parenthesis the average solution times by IR on the instances that could be solved by both of the MIPs. Note that all the instances solved by SR could actually be solved by IR, too.

The figures show that – except for the small instances that can be solved in a matter of seconds by either MIP – the proposed item-related representation outperforms the sequence-related representation both in terms of the number of instances solved to optimality and search time. It solved all the problem instances with $N_I N_T \leq 60$, hence it extended the applicability of exact optimisation methods to instances of industrially relevant size. The advantage of using a more compact formulation with exponentially less binary variables manifested itself especially for large number of items, where IR solved problems in a matter of minutes that were intractable for previous approaches.

The new MIP gave up to 2 orders of magnitude speedup also on instances that were solvable to optimality by both approaches. At the same time, there were 51 instances, all of them with $N_I \leq 5$, whose solution took longer with IR than with SR. This

N_I	N_T	Solved (%)		Time (sec)		
		IR	SR	IR	IR*	SR
3	4	100	100	0.00	(0.00)	0.00
	6	100	100	0.00	(0.00)	0.00
	8	100	100	0.00	(0.00)	0.00
	10	100	100	0.06	(0.06)	0.00
	12	100	100	0.72	(0.72)	0.22
	14	100	100	2.39	(2.39)	1.56
	16	100	100	7.56	(7.56)	7.83
	18	100	100	23.83	(23.83)	26.67
	20	100	100	55.06	(55.06)	95.28
4	4	100	100	0.00	(0.00)	0.00
	6	100	100	0.06	(0.06)	0.00
	8	100	100	0.94	(0.94)	0.83
	10	100	100	2.44	(2.44)	6.72
	12	100	100	13.44	(13.44)	48.33
	14	100	100	50.06	(50.06)	355.56
	16	94	83	330.76	(341.87)	2103.00
	18	89	50	1241.38	(653.89)	2884.44
	20	89	33	2067.94	(364.17)	3135.83
5	4	100	100	0.00	(0.00)	0.00
	6	100	100	0.94	(0.94)	2.00
	8	100	100	5.33	(5.33)	20.72
	10	100	100	30.39	(30.39)	400.56
	12	100	56	292.11	(115.10)	1478.20
	14	89	17	1352.88	(9.00)	224.00
	16	67	6	2593.50	(2.00)	366.00
	18	33	6	2286.83	(8.00)	762.00
	20	33	0	770.17		-
6	4	100	100	0.39	(0.39)	5.00
	6	100	100	4.33	(4.33)	77.39
	8	100	72	22.56	(22.62)	1237.69
	10	100	6	391.17	(546.00)	4881.00
	12	83	0	1523.13		-
	14	39	0	2479.14		-
	16	28	0	1016.00		-
	18	22	0	547.75		-
	20	18	0	2353.33		-

Table 1: Experimental results for 3-6 items. Column *Solved* shows the percentage of instances solved to optimality, while *Time* displays the average solution times for the proposed item-related *(IR)* and the previous sequence-related *(SR)* formulation. Dash '-' means that none of the instances with the given size could be solved in 2 hours.

N_I	N_T	Solved (%)		Time (sec)		
		IR	SR	IR	IR*	SR
7	4	100	94	2.00	(2.06)	192.71
	6	100	17	20.11	(15.33)	1563.33
	8	100	0	196.06		-
	10	89	0	1582.31		-
	12	33	0	372.33		-
	14	28	0	752.20		-
	16	11	0	167.50		-
	18	11	0	579.00		-
	20	11	0	202.50		-
8	4	100	61	7.33	(8.18)	901.27
	6	100	0	62.72		-
	8	100	0	1299.00		-
	10	61	0	2598.36		-
	12	17	0	591.33		-
	14	17	0	2552.67		-
	16	17	0	1806.00		-
9	4	100	11	19.56	(22.50)	1392.00
	6	100	0	272.89		-
	8	83	0	1525.40		-
	10	33	0	2071.50		-
	12	17	0	1580.67		-
	14	17	0	417.33		-
10	4	100	0	67.17		-
	6	100	0	1179.83		-
	8	39	0	1486.57		-
	10	30	0	671.33		-

Table 2: Experimental results for 7-10 items. Column *Solved* shows the percentage of instances solved to optimality, while *Time* displays the average solution times for the proposed item-related *(IR)* and the previous sequence-related *(SR)* formulation. Dash '-' means that none of the instances with the given size could be solved in 2 hours.

difference exceeded 10 seconds in 6 cases, and 1 minute in 2 cases. Where optimal solutions could not be found, the reason of the failure was either a timeout (typically for SR on 5 items or less, and IR) or memory overflow (SR on 6 items or more).

6.2 Results on variants of the problem

We randomly selected 100 problem instances that were solvable in the previous round of experiments to measure the effect of the zero-switch property on the solution process. For 80 of these instances, the optimal solution of the original CLSPSD respected the zero-switch property. For the remaining 20 instances, forcing this property deterio-

N_I	N_T	Solved (%)		Time (sec)		N_I	N_T	Solved (%)		Time (sec)	
		IR	IR^b	IR	IR^b			IR	IR^b	IR	IR^b
3	4	100	100	0.00	0.00	6	4	100	100	0.39	0.50
	6	100	100	0.00	0.00		6	100	100	4.33	4.33
	8	100	100	0.00	0.00		8	100	100	22.56	42.67
	10	100	100	0.06	0.33		10	100	100	391.17	315.17
	12	100	100	0.72	1.17		12	83	50	1523.13	1265.67
	14	100	100	2.39	3.83		14	39	33	2479.14	2461.50
	16	100	100	7.56	6.83		16	28	33	1016.00	618.50
4	4	100	100	0.00	0.00	7	4	100	100	2.00	2.00
	6	100	100	0.06	0.17		6	100	100	20.11	12.33
	8	100	100	0.94	1.00		8	100	100	196.06	282.33
	10	100	100	2.44	6.83		10	89	67	1582.31	1302.50
	12	100	100	13.44	22.50		12	33	50	372.33	1785.00
	14	100	100	50.06	48.33		14	28	33	752.20	1709.00
	16	94	100	330.76	910.33		16	11	0	167.50	-
5	4	100	100	0.00	0.00	8	4	100	100	7.33	8.33
	6	100	100	0.94	1.50		6	100	100	62.72	75.33
	8	100	100	5.33	5.00		8	100	100	1299.00	787.83
	10	100	100	30.39	34.83		10	61	33	2598.36	94.00
	12	100	100	292.11	468.50		12	17	33	591.33	2749.50
	14	89	67	1352.88	412.25		14	17	33	2552.67	1054.00
	16	67	50	2593.50	334.00		16	17	0	1806.00	-

Table 3: Experimental results without and with backlogging. Percentage of instance solved to optimality (in column *Solved*) and average solution times (*Time*) without (*IR*) and with (*IR^b*) backlogging. Dash '-' means that none of the instances with the given size could be solved in 2 hours.

rated the solutions by at most 1.18%. Most often, adding or removing the zero-switch property did not affect the solution time significantly. For 47 instances, the difference in solution time was less than 1 second; of the remaining 52 instances, 29 were solved more quickly without the property, and 24 with the property. In only 5 instances was the difference in solution speed greater than 50%. Hence, we suggest that the zero-switch property should not be enforced in this MIP model of CLSPSD, because it runs a risk of losing optimality without reducing the computational complexity. Note also that these results are representative for the case when the demand values d_t^i are generated using uniform distribution. For different demand profiles, e.g., in case of occasional large demands, the zero-switch property can easily render a problem instance infeasible, or can lead to serious sub-optimality.

In order to measure how the introduction of backlogging affects the complexity of the problem, we created a smaller set of 252 problem instances using the same problem generator. However, capacity utilisation U was now picked from $\{0.7, 0.9\}$, and the time-proportional setup cost coefficient r from $\{50, 200, 500\}$. For each instance, backlogging costs g^i were randomised from $[50, 200]$ with uniform distribution.

The results of the experiments are presented in Table 3, showing the percentage of instances that could be solved to optimality and the average solution times for given combinations of N_I and N_T. Columns *IR* refer to the basic MIP without backlogging, while *IR*^{b} stands for the backlogging case. Contrary to our expectations, the performance difference between the two model variants was not statistically significant: only 3.1% more instances were solved without, than with backlogging. Rather surprisingly, the overall average solution time was 24.5% lower with backlogging than without it. However, this series of experiments illustrates that the proposed approach adapts well to different variants and extensions of the basic lot-sizing model.

7 Conclusions and future work

This paper addressed the capacitated lot-sizing and scheduling problem with sequence-dependent setup times and costs (CLSPSD). We showed that the complexity of this large-bucket lot-sizing problem originates from the series of implicit sequencing problems that have to be solved for the items produced in each time period. We presented an efficient algorithm for determining during pre-processing all item sequences that could appear in an optimal solution. We introduced a novel MIP formulation of CLSPSD that relies on a compact representation of those sequences by using item-related binary variables. To ensure the MIP is correct, we added constraints which bound from below the setup costs for each time period, based on the efficient sequences, and we presented a heuristic algorithm for generating coefficients of these constraints which give tight LP relaxations. Given these new constraints and the compact model, the proposed MIP outperforms all previously known optimisation approaches: it solves problems with orders of magnitude speedup, and can solve instances of industrially relevant size.

As one may expect, the stochastic variant of this problem is much more complex, but has still attracted attention in the literature due to its importance from both theoretical and practical points of view (see Kämpf and Köchel, 2006). Our future work will focus on extending these results to the stochastic version of the same problem, where demands are described by (not necessarily independent) random variables with known probability density functions. Departing from the MIP proposed in this paper, we defined a stochastic program in the Stochastic OPL Language (Tarim et al., 2006). Currently, we are experimenting with solving this stochastic program for instances with different sizes and characteristics, using different scenario reduction techniques. Our aim is to extend the applicability of this approach to realistic problem sizes.

Acknowledgements

This work has been supported by Science Foundation Ireland under Grant Nos. 00/PI.1/C075 and 03/CE3/I405, and partly by the VITAL NKFP grant No. 2/010/2004. A. Kovács acknowledges the support of the ERCIM 'Alain Bensoussan' fellowship programme and the János Bolyai scholarship of the Hungarian Academy of Sciences.

References

Bellman, R., 1962. Dynamic Programming Treatment of the Travelling Salesman Problem. Journal of the ACM 9(1), 61–63.

Belvaux, G., Wolsey, L.A., 2001. Modelling Practical Lot-Sizing Problems as Mixed-integer Programs. Management Science 47(7), 993–1007.

Chen, W.H., Thizy, J.M., 1990. Analysis of Relaxations for the Multi-item Capacitated Lot-sizing Problem. Annals of Operations Research 26, 29–72.

Drexl, A., Kimms, A., 1997. Lot-sizing and Scheduling – Survey and Extensions. European Journal of Operational Research 99, 221–235.

Gupta, D., Magnusson, T., 2005. The Capacitated Lot-sizing and Scheduling Problem with Sequence-dependent Setup Costs and Setup Times. Computers & Operations Research 32, 727–747.

Haase, K., Kimms, A., 2000. Lot Sizing and Scheduling with Sequence-dependent Setup Costs and Times and Efficient Rescheduling Opportunities. International Journal of Production Economics 66(2), 159–169.

Kämpf, M., Köchel, P., 2006. Simulation-based Sequencing and Lot Size Optimisation for a Production-and-inventory System with Multiple Items. International Journal of Production Economics 104(1), 191-200.

Karimi, B., Fatemi Ghomi, S.M.T., Wilson, J.M., 2003. The Capacitated Lot-sizing Problem: A Review of Models and Algorithms. Omega 31, 365–378.

Marchand, H., Martin, A., Weismantel, R., Wolsey L., 2002. Cutting Planes in Integer and Mixed Integer Programming. Discrete Applied Mathematics 123(1-3), 397–446.

Meyr, H., 2000. Simultaneous Lotsizing and Scheduling by Combining Local Search with Dual Reoptimization. European Journal of Operational Research 120, 311–326.

Meyr, H., 2002. Simultaneous Lotsizing and Scheduling on Parallel Machines. European Journal of Operational Research 139, 277–292.

Tarim, S.A., Manandhar, S., Walsh, T., 2006. Stochastic Constraint Programming: A Scenario-based Approach. Constraints 11, 53–80.

Williams, H.P., 1999. Model Building in Mathematical Programming, 4th Edition, Wiley.