Single Machine Multi-product Capacitated Lotsizing with Sequence-dependent Setups

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0568.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	06-Oct-2006
Complete List of Authors:	Almada-Lobo, Bernardo; University of Porto, Faculty of Engineering; Massachusetts Institute of Technology, Operations Research Center Klabjan, Diego; University of Illinois at Urbana-Champaign, Department of Mechanical and Industrial Engineering Carravilla, Maria Antónia; University of Porto, Faculty of Engineering Oliveira, Jose; University of Porto, Faculty of Engineering
Keywords:	LOT SIZING, INTEGER PROGRAMMING, HEURISTICS
Keywords (user):	Sequence-dependent setup, capacitated lotsizing and scheduling

Single Machine Multi-product Capacitated Lotsizing with Sequence-dependent Setups

Bernardo Almada-Lobo[a,b,*] Diego Klabjan[b] Maria Antónia Carravilla[a,c]
José F. Oliveira[a,c]

[a] Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias s/n, Porto 4200-465, Portugal

[b] Massachusetts Institute of Technology, Cambridge, MA, USA

[c] Instituto de Engenharia de Sistemas e Computadores do Porto Campus da FEUP, Rua Dr. Roberto Frias 378, Porto 4200-465, Portugal

Abstract

In production planning in the glass container industry, machine dependent setup times and costs are incurred for switchovers from one product to another. The resulting multi-item capacitated lot sizing problem has sequence-dependent setup times and costs. We present two novel linear mixed integer programming formulations for this problem, incorporating all the necessary features of setup carryovers. The compact formulation has polynomially many constraints, while, on the other hand, the stronger formulation uses an exponential number of constraints that can be separated in polynomial time. We also present a five-step heuristic, which is effective both in finding a feasible solution (even for tightly capacitated instances) and in producing good solutions to these problems. We report computational experiments.

Keywords: capacitated lotsizing and scheduling, sequence-dependent setup, integer programming, heuristics

1 Introduction

Production planning arising in the glass container industry is a complex process. It is a semi-continuous manufacturing process, which includes the glass production (the continuous part) and the container manufacturing (the discrete part). Furnaces are operated continuously and machine lines operate on a twenty four hour seven days a week basis. During a product changeover on a molding machine, the furnace keeps feeding the machine with molten glass, however, the gobs are discarded and melted down again in the furnace (wasting a huge amount of energy, the main production cost). Fast job changes with fewer job change parts lower the energy costs. Therefore, there is a sequence dependent setup time (and proportional cost) in a product changeover. Since machine setups consume the furnace capacity, it is essential to carry the setup from one period to the next. This production planning problem results in a capacitated multi-item deterministic lotsizing problem.

The lotsizing problem is to find production orders or lots in order to satisfy customer demand. The objective is to minimize holding and setup costs. This problem has been intensively studied

*Corresponding author. *E-mail address*: almada.lobo@fe.up.pt; tel: +351225082133

and, therefore, many models were proposed to describe it, involving different features and assumptions. The majority of the models consider a finite planning horizon divided into discrete time periods. The single stage, multi-item lotsizing problem with capacity constraints is referred to as the capacitated lotsizing problem (CLSP). CLSP is considered to be a large-bucket problem, because several products/setups may be produced/performed per period. On the other hand, in small-bucket problems, at most one setup may be executed during a period. Different models are distinguished with respect to the number of products that can be produced within a period and to the period capacity utilization (all-or-nothing assumption or partial). All these models determine simultaneously the size and sequence of lots. Despite allowing more precise plans, sub-dividing the (macro-) periods of CLSP into several (micro-) periods increases the complexity of these mixed integer linear programming models. Drexl and Kimms [1997] present and explain the main differences of these formal models.

We study single machine CLSP with sequence dependent setup times and costs. In each time period several products can be produced. Whenever a switch of a product is performed, a certain amount of time is consumed and the setup cost is incurred. This time and cost depend on the two products. In each time period at most a given amount of time can be allotted to switchover times and to the actual production time. This leads to sequence dependent setup times and costs. A further complicating matter is in the fact that a product can be set up at the end of a time period and the actual production starts in the next time period. We call this property setup carryover. We present two novel linear mixed integer programming formulations for single machine CLSP with sequence-dependent setup times and costs, incorporating all the necessary features of setup carryovers. The compact formulation has a polynomial number of constraints. The efficiency of this model is benchmarked against the one proposed by Gupta and Magnusson [2005], which is actually not a completely accurate formulation (Almada-Lobo et al. [2006]). The second formulation uses an exponential number of constraints, which makes it impractical to be solved directly. We generate constraints dynamically by presenting a polynomial time separation algorithm. We compare the strength of the linear programming relaxations of both formulations. The two formulations are improved by adding valid inequalities that lead to good lower bounds. Since this problem is NP-hard, we also present a five-step heuristic to solve large problem instances. The flexibility of the heuristic's first two steps allows us to find feasible solutions even under tightly capacitated scenarios. The last three steps focus on solution quality. We compare the heuristic solution against the lower bound generated by the strongest model and against other known heuristics. The main contributions of our work are as follows. To the best of our knowledge, the two presented formulations are the first exact formulations for the problem under consideration. We also formally prove a relationship between the two linear programming relaxations. Another major contribution of our work is the underlying heuristic.

In Section 2 we first present the compact formulation of the problem, which is also benchmarked against the model presented by Gupta and Magnusson [2005]. We then propose the stronger formulation and prove that it provides a better linear programming relaxation. In Section 3 we explain the heuristic for finding good solutions. Numerical experiments are given in Section 4. We conclude this introduction with a literature review.

Literature Review

Standard CLSP does not sequence or schedule products within a period. In addition, it assumes that setup costs occur for each lot in a period, even if the last product to be produced in a period

is the first one in the period that follows. For manufacturing environments with considerable setup times with respect to the length of a period and with tight capacity, which is the case in the glass container industry, this model may not provide any feasible production plans. Therefore, setup carryovers need to be incorporated in CLSP. The setup carryover occurs when a product is the last one to be produced in a period and the first one in the following period. In this case no setup is required in the latter period. Moreover, setups are preserved over idle periods and an idle time at the end of a period may be used to perform a setup for the first product to be produced at the beginning of the next period.

Despite CLSP being intensively studied, modeling setup carryover in CLSP has not received much attention due to model complexity and computational difficulty, Sox and Gao [1999]. The efficiency of solving CLSP depends on the structure of setups. Different studies have demonstrated that proper accounting for setup times and carryovers decreases the number of setups and also frees a significant amount of production capacity, Porkka et al. [2003]. Gopalakrishnan et al. [1995] developed a modeling framework for formulating CLSP with sequence and product independent setup times and setup carryovers. In their model, a fixed charge is incurred whenever a product is produced in a period. The authors conclude that the model complexity could be reduced by looking for alternative ways to model setup carryover. In Gopalakrishnan [2000], a modified framework for modeling setup carryover in CLSP is presented, incorporating sequence independent and product dependent setup times and costs. Nevertheless, setup carryover is modeled in a similar way as in Gopalakrishnan et al. [1995] and decision variables have the same interpretation. Sox and Gao [1999] present a more efficient mixed integer linear programming model to CLSP with sequence independent setup costs and no setup times. Kang et al. [1999] propose a different approach, consisting of breaking the entire schedule into smaller segments, called split-sequences. A drawback of this model is that the number of split-sequences L_t in period t is a parameter. Therefore, multiple runs with different values for L_t are needed for optimization. Moreover, setup times are not considered in this model. Porkka et al. [2003] modify the mixed integer linear programming based setup carryover model of Sox and Gao [1999] by using sequence independent setup times. To avoid the complexity of separate setup costs and times, explicit setup costs are excluded from the model. Suerie and Stadtler [2003] present a new model formulation for CLSP with sequence independent setup costs and times. The authors start with the standard CLSP model, use the standard facility location reformulation, and introduce new sets of variables and constraints to model the setup carryover. Gopalakrishnan et al. [2001] also address CLSP with setup carryover and present two effective tabu-search heuristics.

All the aforementioned manuscripts address CLSP with constant sequence independent setup times and/or setup costs, with a setup carryover. In fact, these models do not consider lot sequencing within a period. They focus only on determining the products produced last and first in two consecutive periods, and also the configuration of the machine at the end of the period. Due to the "correct" calculation of the setup costs by linking lots of adjacent periods, these models are usually referred to as CLSP with linked lot-sizes (Haase [1994]).

Notwithstanding, there are several examples of process industries in which sequence-dependent setup times and costs are considerable, such as glass container and some chemical industries. Clark and Clark [2000] model CLSP with sequence-dependent setup times using a new mixed integer programming formulation. They assume that exactly a given number of setups occur in a time period between any two given products, independently of their demand patterns. Haase and Kimms [2000] propose a model for CLSP with sequence dependent setup times and costs in which the efficient product sequences are pre-determined. Therefore, it is to decide which sequences will be

used in each period. The authors also assume that the inventory of an item must be null at the beginning of a period to allow its production in the period. Gupta and Magnusson [2005] extended the framework proposed by Gopalakrishnan [2000] to incorporate sequence dependent setup times and setup costs. The setup carryover is modeled in the same way as in Gopalakrishnan et al. [1995] and Gopalakrishnan [2000].

2 New Models for CLSP with Sequence-dependent Setup Costs and Times

2.1 The First Formulation

Throughout the exposition, t denotes time periods, which range from 1 to T, and i and j index products, which are labeled from 1 to N. We denote by $[M]$ the set $\{1, 2, \ldots, M\}$. We consider a general single-stage model involving multiple items to be scheduled on a single machine, whose data consist of:

h_i	cost of carrying one unit of stock of product i from one period to the next,
p_i	processing time of one unit of product i,
d_{it}	demand for product i at the end of period t,
C_t	capacity of the machine in period t (measured in time units),
s_{ij}	time needed to set up the machine from product i to product j,
c_{ij}	cost incurred to set up the machine from product i to product j.

In addition, let $M_{it} = \min\left\{\frac{C_t}{p_i}, \sum_{u=t}^{T} d_{iu}\right\}$ be an upper bound on the quantity of product i to be produced in period t.

The total cost consists of the holding cost and the machine setup cost. We do not allow stockouts, which is common in the deterministic demand setting. We assume that the triangle inequality with respect to the setup cost and time holds, i.e., $c_{ik} \leq c_{ij} + c_{jk}$ and $s_{ik} \leq s_{ij} + s_{jk}$ for all products i, j, and k. This assumption holds in many practical settings, and definitely in our glass container case.

In order to capture the lotsizes and the resulting inventory we need the following decision variables:

X_{it}	quantity of product i produced in period t,
I_{it}	stock of product i at the end of period t.

In order to capture the setup cost and time, we introduce

$$T_{ijt} = \begin{cases} 1 & \text{if a setup occurs on the machine configuration state from} \\ & \quad \text{product } i \text{ to product } j \text{ in period } t, \\ 0 & \text{otherwise.} \end{cases}$$

We assume that $T_{iit} = 0$ for every product i. Recall that we allow the machine to be set up with product i at the end of a time period and then start the next time period with the same product but not occurring any extra setup cost. Since such a setting involves two consecutive time periods, a new decision variable needs to be introduced. Let

$$\alpha_{it} = \begin{cases} 1 & \text{if the machine is set up for product } i \text{ at the beginning of period } t, \\ 0 & \text{otherwise.} \end{cases}$$

We also need the auxiliary variables V_{it} that assign product i in period t.

Our first formulation (F_1) for CLSP with sequence-dependent setup costs and times, and setup carryover is as follows:

$$\min \sum_i \sum_j \sum_t c_{ij} \cdot T_{ijt} + \sum_i \sum_t h_i \cdot I_{it} \tag{1}$$

$$I_{it} = I_{i(t-1)} + X_{it} - d_{it} \qquad i \in [N], t \in [T] \tag{2}$$

$$\sum_i p_i \cdot X_{it} + \sum_i \sum_j \sum_t s_{ij} \cdot T_{ijt} \leq C_t \qquad t \in [T] \tag{3}$$

$$X_{it} \leq M_{it} \cdot \left(\sum_j T_{jit} + \alpha_{it} \right) \quad i \in [N], t \in [T] \tag{4}$$

$$\sum_i \alpha_{it} = 1 \qquad t \in [T] \tag{5}$$

$$\alpha_{it} + \sum_j T_{jit} = \alpha_{i(t+1)} + \sum_j T_{ijt} \qquad i \in [N], t \in [T] \tag{6}$$

$$V_{it} + N \cdot T_{ijt} - (N-1) - N \cdot \alpha_{it} \leq V_{jt} \qquad \begin{array}{l} i \in [N], j \in [N] \setminus \{i\}, \\ t \in [T] \end{array} \tag{7}$$

$$(X_{it}, I_{it}, \alpha_{it}, V_{it}) \geq 0, T_{ijt} \in \{0, 1\}. \tag{8}$$

The objective function (1) minimizes the sum of sequence-dependent setup costs and the holding cost. Constraints (2) represent the inventory balances and (3) ensure that the total production and setup time in each period does not exceed the available capacity. Requirement (4) guarantees that a product is produced only if the machine has been set up for it. Constraints (5)-(7) determine the sequence of products on the machine in each period and keep track of the machine configuration state by recording the product that a machine is ready to process (setup carryover information is thereby tracked). We next discuss these constraints further. Hereafter we refer to T_{jit} as an input setup for product i in period t and to T_{ijt} as an output setup for product i in period t.

The model does not explicitly have binary variables representing whether a product is produced or not in a certain period. Production of product i can occur in period t if the machine is set up for i at the beginning of t or if at least one input setup is performed for product i. These conditions are guaranteed by constraints (4).

Constraints (5) ensure that α_{it} is one for exactly one product in a given period, i.e., that the machine is set up for exactly one product at the beginning of each period.

Constraints (6) ensure a balanced network flow of the machine configuration states and carry the setup configuration state of the machine into the next period. If no setup is performed in period t, which happens when $T_{ijt} = 0$ for every i and j, configuration state of the machine is carried from period $t-1$ to period $t+1$. The idle time of a period t may also be used to perform an empty setup for the first product to be produced at the beginning of the next period. If there is an input setup and no output one for product i in period t, it means this setup was the last one to be performed on the machine in period t and, consequently, the machine is configured for product i at the beginning of the next period, i.e., $\alpha_{i(t+1)} = 1$. On the other hand, if there is an output

setup and no input one, it means that the machine is configured for product i at the beginning of period t, i.e., $\alpha_{it} = 1$.

In general, constraints (6) impose that flow in equals flow out. We note that constraints (2)-(6) and (8) allow solutions with cycles that are not disjoint. However, such a scenario will never occur in an optimal solution since it is easy to see that the triangle inequality assumption implies that the in flow of every node is at most 1.

Let us define a digraph $G = ([N], A = [N] \times [N])$, where nodes correspond to products and an arc $a = (i, j)$ corresponds to the setup from product i to product j. It follows that the set $\{(i, j)|T_{ijt} = 1\}$ corresponds to a collection of disjoint paths and cycles. Constraints (2)-(6) and (8) admit solutions that have at most one path, but potentially many cycles, called also subtours, see Figure 1. The left configuration represents a solution that entails one path from product i_1 to product i_4 and 3 disconnected subtours. The right configuration illustrates a solution with a cycle that starts and ends in product i_1 and 2 disconnected subtours.

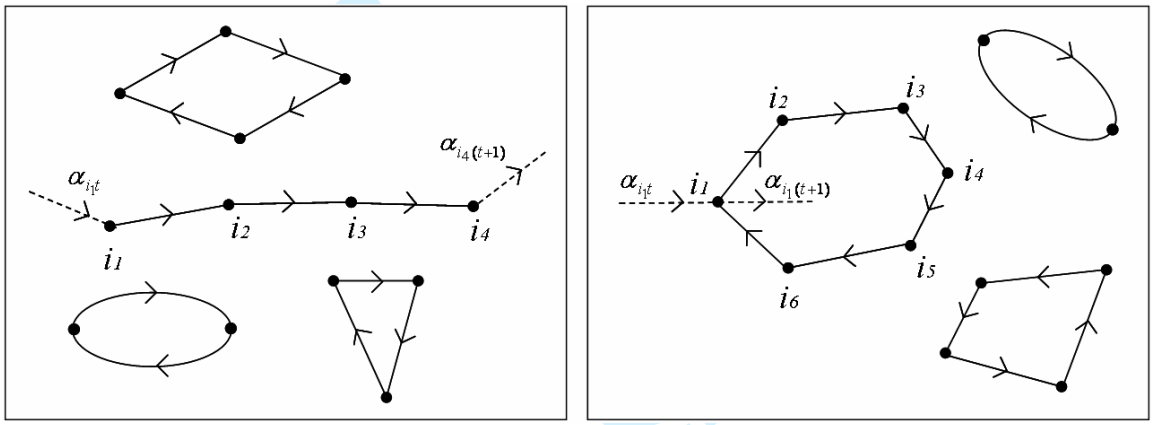


Figure 1: Possible configurations

In order to obtain a feasible solution, we must impose that we have only a single connected component linked to α's. It can be shown that constraints (7) achieve this by using auxiliary variables that value the machine state through any sequence. It is also not difficult to see that no feasible solution is excluded by (7). A similar modeling trick is known in the context of the traveling salesman problem, see, e.g., Nemhauser and Wolsey [1988].

We conclude that the underlaying subgraph associated with an optimal T is either a cycle or a path. Thus it looks like the one depicted in Figure 1 except that there are no disconnected subtours.

Note that the integrality condition of α_{it} is not necessary. Let us assume that $\alpha_{i1} = 1$. From (6) it follows that α_{j2} is an integer for every j. Together with (5) we obtain that α_{j2} is binary. This reasoning rolls over to the subsequent periods. If the machine is not set up for any product in the first time period, then the integrality requirements of α_{j1}'s need to be added. As an alternative to avoid it, the machine can be set up for a virtual product at the beginning of the planning period. No setup times and costs would be incurred if a setup occurs between this product and any other one.

The presented model assumes the single-machine case. The extension of this model to multiple machines is straightforward, also considering other usual features, such as shortages, backlogging

costs and lower and upper bounds on lot sizes.

Example 1. Consider the following small data set provided in Table 1 for a three-product, three-period problem.

Table 1: Data for the three product, three-period problem

	d_{it}			c_{ij}			
	$t=1$	$t=2$	$t=3$	$j=1$	$j=2$	$j=3$	h_i
$i=1$	15	5	10	0	3	3	10
$i=2$	20	35	20	4	0	3	15
$i=3$	0	110	40	5	5	0	20

Setup times are $s_{ij} = 5$ time units for $i \neq j$ and zero otherwise. Moreover, $C_t = 100$ for every t and $p_i = 1$ for every i.

We solved the model by using a commercial solver. Figures 2 and 3 show the optimal solution for this instance, when the machine is set up for product 3 at the beginning of the planning period, i.e., $\alpha_{31} = 1$. Note that there is idle time at the end of period 3.

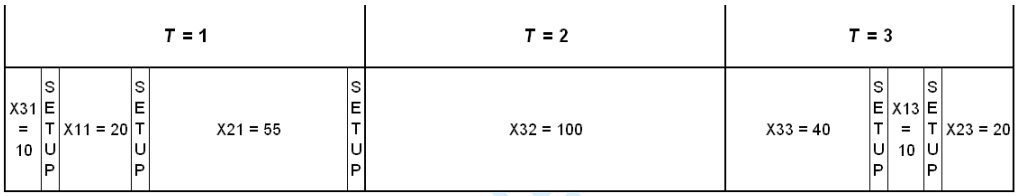


Figure 2: Gantt chart of the optimal solution

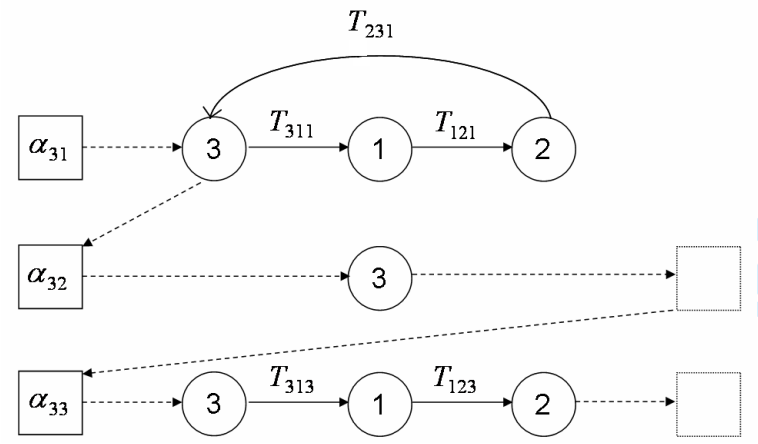


Figure 3: Network representation of the optimal solution

It entails seven productions and five setups. The last setup in period 1 is an empty one, since the idle time at the end of that period is used to set up for product 3, the first one to be produced

at the beginning of period 2. In period 2 only product 3 is produced and no setup is performed. Therefore, the setup state of the machine for product 3 is carried from period 1 to period 3. At the end of the planning horizon the machine is ready to produce product 2. □

2.1.1 Benchmarking the First Formulation

In this section we compare our model F_1 against Gupta and Magnusson's model. The experiments were conducted on an IBM machine with a 3.2 GHz processor and 1GB of random access memory and CPLEX 9.0 as a mixed integer programming solver.

In Table 2 we conduct a benchmark on the complexity (in terms of the number of binary variables, continuous variables and constraints) of F_1 against the models presented in Clark and Clark [2000] and Gupta and Magnusson [2005]. Here S is the maximum allowable number of setups per period used in Clark and Clark's model.

Table 2: Benchmark of models' complexity

Model	Binary variables	Continuous variables	Constraints
F_1	N^2T	$4NT$	$NT(2+N)+2T$
Clark and Clark [2000]	$N^2(ST-1)+N$	SNT	$NT(SN+2S+2)$ $+1-N$
Gupta and Magnusson [2005]	N^2T+4NT	$2NT$	$NT(5+3N)+8T$

As we can observe from the number of binary variables, F_1 requires less variables than the other two. In Example 1, F_1 has 27 binary variables, whilst the Clark and Clark's model has 75 (S has to be set to 3 to reach the same optimal solution) and the Gupta and Magnusson's model has 63 binary variables.

In order to computationally evaluate the models, we generated random instances based on the parameters used by Haase and Kimms [2000]. The machine is always set up for product 1 at the beginning of the planning horizon. The number of products N ranges from 3 to 10, the number of periods were 3, 4, 5, 7, 10, and 15, and the capacity utilization (Cut) defined as $\sum_i d_{it}/C_t$ takes the value 0.6. Processing times (p_i) are unitary for all of the products. Demand (d_{it}) is selected based on the uniform distribution $U[40,60]$, holding costs (h_i) based on $U[2,10]$, and setup times (s_{ij}) based on $U[5,10]$ (the triangle inequalities hold). The setup costs c_{ij} are proportional to the setup times, i.e., $c_{ij} = \theta \cdot s_{ij}$ with $\theta = 50$. Note that θ implicity defines a relationship between the setup and holding costs. In the case of the company producing glass containers that inspired this work, θ falls in the interval $[50,75]$.

For each triplet N, T, and Cut, three different instances were generated and the average running times (in seconds) were calculated. Table 3 gives the computational times to obtain an optimal solution by model F_1. An empty field means that at least one of the instances could not be solved optimally within a 3600 seconds limit.

A comparison between the efficiency of F_1 and the Gupta and Magnusson's model (denoted by GM) for $Cut = 0.6$ is shown in Figure 4. We did not compare F_1 to the Clark and Clark's model since multiple runs with different values for S would be needed for reaching the same optimal solution. The vertical axis denoted by F_1/GM represents the ratio of the respective computational times.

Table 3: Average running times for $Cut = 0.6$ and $\theta = 50$ (model F_1)

	T					
N	3	4	5	7	10	15
3	0.1	0.1	0.1	0.2	0.2	14.7
4	0.1	0.1	0.3	2.8	1.3	34.7
5	0.2	0.3	0.7	7.3	7.9	300.1
6	0.2	0.3	0.7	2.3	24.5	593.6
7	0.5	1.2	2.0	5.1	310.7	1628.0
8	0.7	1.1	4.7	37.7	162.9	
9	0.7	2.1	9.4	55.5	967.8	
10	1.4	1.2	39.7	313.1	2875.7	

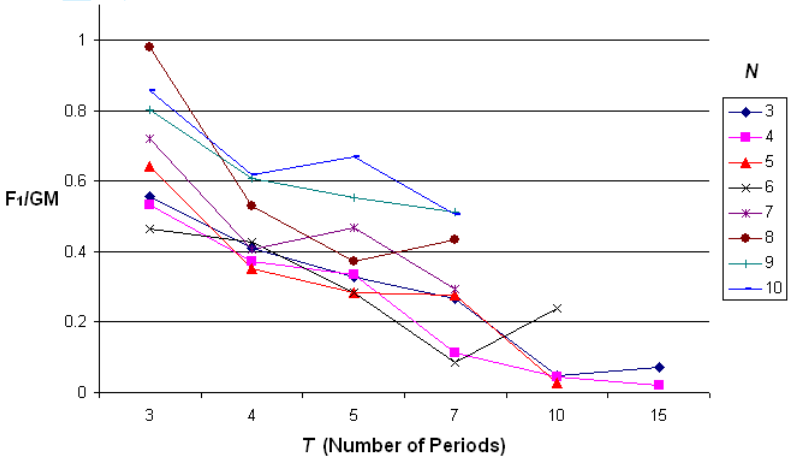


Figure 4: Comparison of run-time performance between F_1 and GM models for $Cut = 0.6$

It is clear that F_1 model is always more efficient than the GM model. Moreover, as the number of periods increases, the ratio F_1/GM tends to 0, i.e., the performance of F_1 is significantly better with increasing T. It turns out that the number of products does not influence significantly the relative efficiency.

2.2 The Compact Formulation

In this section we propose an alternative equivalent formulation (F_2) to CLSP with sequence dependent setup times and setup costs.

Since the triangle inequality holds for both setup costs and setup times, constraints

$$\sum_j T_{ijt} \leq 1 \qquad i \in [N], t \in [T] \tag{9}$$

$$\sum_j T_{jit} \leq 1 \qquad i \in [N], t \in [T] \tag{10}$$

are always satisfied in an optimal solution. The following inequalities are an alternative formulation to constraints (7):

$$\vartheta_{it} = \alpha_{i(t+1)} + \sum_j T_{ijt} \qquad i \in [N], t \in [T] \tag{11}$$

$$\sum_{i\in S}\sum_{j\in S} T_{ijt} \leq \sum_{i\in S} \vartheta_{it} - \vartheta_{kt} + \alpha_{kt} \qquad t \in [T], k \in S, S \subseteq [N]. \tag{12}$$

By substituting (11) in (12), we obtain the equivalent requirement

$$\sum_{i\in S}\sum_{j\notin S} T_{ijt} + \sum_{i\in S} \alpha_{i(t+1)} \geq \sum_j T_{jkt} \qquad t \in [T], k \in S, S \subseteq [N]. \tag{13}$$

If there is a subtour on a subset S of nodes, the left-hand side of (12) equals to $|S|$. If $\alpha_{it} = 0$ for every $i \in S$, this constraint is violated since its right-hand side equals to $|S| - 1$ for every product k in the cycle. Note that $\vartheta_{it} = 1$ for every $i \in S$.

Now we argue that a feasible solution is not excluded by (13). To this end, let (X, I, α, T) be feasible to F_1. It suffices to consider the case in which the right-hand side of (13) equals one. For a fixed S and $k \in S$, let $h \in [N]$ be such that $T_{hkt} = 1$. We distinguish two cases: $h \in S$ or $h \notin S$.

$h \in S$: If subset S contains the node set of the entire cycle or path corresponding to the solution, then $\sum_{i\in S} \alpha_{i(t+1)} = 1$ and (13) is satisfied. On the other hand, if this is not the case, then we have to consider three different scenarios.

(a) If the solution entails a cycle, then clearly $\sum_{i\in S}\sum_{j\notin S} T_{ijt} \geq 1$ since the cycle must "enter" and "leave" S.

(b) If the solution consists of a path that ends at node k, then $\alpha_{k(t+1)} = 1$ and, therefore, $\sum_{i\in S} \alpha_{i(t+1)} = 1$.

(c) If the solution consists of a path that does not end at node k, then either the path ends at another node in S (and $\sum_{i\in S} \alpha_{i(t+1)} = 1$) or it ends at a node not in S and, in this case, $\sum_{i\in S}\sum_{j\notin S} T_{ijt} \geq 1$.

Each of these scenarios yields constraint (13).

$h \notin S$: This case is similar to the above case.

The F_2 formulation reads

$$\begin{aligned} \min & \sum_i \sum_j \sum_t c_{ij} \cdot T_{ijt} + \sum_i \sum_t h_i \cdot I_{it} \\ & \text{subject to } (2) - (6), (9), (10), (13) \\ & (X_{it}, I_{it}, \alpha_{it}) \geq 0, T_{ijt} \in \{0, 1\}. \end{aligned}$$

Let P_{F_1} and P_{F_2} denote the feasible sets of the linear relaxations of the formulations F_1 and F_2, respectively. In the following theorem we show that F_2 is at least as strong as F_1.

Theorem 1. *We have* $P_{F_2} \subseteq \text{proj}(P_{F_1})$*, where* proj *denotes the projection of* P_{F_1} *on the space of* $(X_{it}, I_{it}, T_{ijt}, \alpha_{it})$.

Proof. We show the statement by exhibiting the appropriate $V's$. They are defined as shortest path distances with respect to predefined weights. Since shortest path distances are well defined on networks with no negative weight cycles, we rely on (12) and appropriately selected weights to show this.

Let $(X^*_{it}, I^*_{it}, T^*_{ijt}, \alpha^*_{it}) \in P_{F_2}$ be arbitrary. We find V^*_{it} such that (7) hold, i.e.,

$$(X^*_{it}, I^*_{it}, T^*_{ijt}, \alpha^*_{it}, V^*_{it}) \in P_{F_1}\,.$$

Consider time period t and let $w_{ij} = (N-1) + N \cdot \alpha^*_{it} - N \cdot T^*_{ijt}$ for every i, j. Consider network G defined in Section 2.1, where each arc has weight w_{ij}. Let us fix an arbitrary source node s and let $dist_i$ denote the length of the shortest path from the source node to node i. We show later that these distances are well defined. They satisfy the following optimality conditions:

$$dist_i + w_{ij} \geq dist_j \qquad (i,j) \in A(G).$$

Then, by setting $\overline{V}_{it} = -dist_i$, we obtain

$$\overline{V}_{it} - (N-1) - N \cdot \alpha^*_{it} + N \cdot T^*_{ijt} \leq \overline{V}_{jt}. \tag{14}$$

To ensure that V_{it} are non-negative, we define $V^*_{it} = \overline{V}_{it} + \max_{j,\bar{t}} |\overline{V}_{j\bar{t}}|$. Clearly, from (14) it follows that (V^*_{ijt}, T^*_{ijt}) satisfy (7). We conclude that $(X^*_{it}, I^*_{it}, T^*_{ijt}, \alpha^*_{it}, V^*_{it}) \in P_{F_1}$.

In order to complete the proof, we need to show that $dist_i$ are well defined. This is equivalent to showing that there are no negative cost cycles with respect to w (see, e.g., Ahuja et al. [1993]). We need to show that $\sum_{(i,j)\in C} w_{ij} \geq 0$ for any cycle C. Let S be the node set of cycle C and $k \in S$. From (12) and nonnegativity of T^*, we have

$$\sum_{(i,j)\in C} T^*_{ijt} \leq \sum_{i,j\in S} T^*_{ijt} \leq \sum_{i\in S} \vartheta_{it} - \vartheta_{kt} + \alpha^*_{kt}. \tag{15}$$

Since

$$\sum_{(i,j)\in C} w_{ij} = (N-1)\cdot|S| + N \cdot \sum_{i\in S} \alpha^*_{it} - N \cdot \sum_{(i,j)\in C} T^*_{ijt} \tag{16}$$

and by using (15), we obtain

$$\sum_{(i,j)\in C} w_{ij} \geq (N-1)\cdot|S| + N \cdot \sum_{i\in S} \alpha^*_{it} - N \cdot \sum_{i\in S} \vartheta_{it} + N \cdot \vartheta_{ik} - N \cdot \alpha^*_{ik}. \tag{17}$$

Replacing (11) into (17), yields

$$\sum_{(i,j)\in C} w_{ij} \geq (N-1)\cdot|S| - N \cdot \sum_{i\in S\setminus\{k\}} \sum_{j} T^*_{jit}.$$

Taking into account (10), leads to

$$\sum_{(i,j)\in C} w_{ij} \geq (N-1)\cdot|S| - N \cdot (|S|-1) \geq 0,$$

since $|S| \leq N$. □

Note that F_2 has an exponential number of constraints. To introduce the most violated (t, S, k) inequalities (13) we need to solve the separation problem. We next show how to solve this problem.

Let $(X^*_{it}, I^*_{it}, T^*_{ijt}, \alpha^*_{it})$ satisfy (2) − (6), (9) and (10). Consider network G with capacity T^*_{ijt} on arc (i, j) in A. Next we define a new network G^0 by adding sink node z to G and arcs (i, z) for every product i with capacity $\alpha^*_{i(t+1)}$. Finally, we solve the min (s, z) cut problem for every product s. The algorithm hinges on the fact that the value of an $s - z$ cut S in G^0 equals to $\sum_{i\in S}\sum_{j\notin S} T^*_{ijt} + \sum_{i\in S} \alpha^*_{i(t+1)}$. If there exists a $k \in S$ such that $\sum_j T^*_{jkt} > Z^*$ with Z^* being the value of the minimum $s - z$ cut, then (t, S, k) inequality is violated.

Since finding a minimum cut in a directed graph can be performed in polynomial time, this separation problem is polynomial.

2.2.1 Valid Inequalities

In this section we present two classes of valid inequalities (cuts) that strengthen F_2.

The first class of valid inequalities is not a family in the original space, but we need to consider a higher dimensional space. Let W_t denote a binary variable that equals 0 if at least one setup is performed in period t and 1 otherwise. Then, the following sets of valid inequalities, denoted as (W_t) inequalities,

$$\begin{aligned}
\alpha_{it} &\le \sum_j T_{ijt} + W_t && i \in [N], t \in [T] \\
\alpha_{it} &\le \sum_j T_{ji(t-1)} + W_{t-1} && i \in [N], t \in [T] \\
1 - W_t &\le \sum_{kj} T_{kjt} && t \in [T] \\
W_t &\le 1 - \frac{\sum_{kj} T_{kjt}}{N} && t \in [T]
\end{aligned}$$

are valid.

To argue that they are valid, it suffices to consider the case $\alpha_{it} = 1$. We distinguish the following four cases.

$\alpha_{i(t-1)} = 0, \alpha_{i(t+1)} = 0$**:** In this case $\sum_j T_{ji(t-1)} \ge 1, \sum_j T_{ijt} \ge 1$, and $W_t = W_{t-1} = 0$ and therefore the inequalities are satisfied.

$\alpha_{i(t-1)} = 1, \alpha_{i(t+1)} = 0$**:** In this case $\sum_j T_{ijt} \ge 1$ and $W_t = 0$. If $\sum_{kj} T_{kj(t-1)} \ge 1$, then also $\sum_j T_{ji(t-1)} \ge 1$ (and $W_{t-1} = 0$) and they are valid. On the other hand, if $\sum_{kj} T_{kj(t-1)} = 0$, then $W_{t-1} = 1$ and again they are valid.

$\alpha_{i(t-1)} = 0, \alpha_{i(t+1)} = 1$**:** This case is similar to the above.

$\alpha_{i(t-1)} = 1, \alpha_{i(t+1)} = 1$**:** It is easy to adapt the second case to this case.

The introduction of W_t does not increase significantly the computational time since there are only T of them, and, furthermore, their integrality is implied.

It is also helpful to use the well known (l, S) inequalities for uncapacitated single-item lotsizing (Barany et al. [1984]). The (l, S) inequalities adjusted for the multi-item case of our problem are expressed as

$$d_{i1l} \leq \sum_{t \in [l] \setminus S} d_{itl} \cdot \left(\sum_j T_{jit} + \alpha_{it} \right) + \sum_{t \in S} X_{it} \qquad l \leq T, S \subseteq [l], \tag{18}$$

where $d_{ipg} = \sum_{t=p}^{g} d_{it}$. Since there are an exponential number of these constraints, we introduce them as cutting planes (instead of adding all of them a priori). To find the most violated (l, S) inequalities, we use the separation algorithm introduced by Barany et al. [1984], which requires $O(NT^2)$ time steps.

3 A Heuristic for CLSP

In this section we propose a heuristic for finding good solutions to CLSP, i.e., upper bounds on the optimal value. The heuristic consists of five basic steps. The first two attempt to find an initial feasible solution, whilst the last three are geared toward improving the quality of the solution. In the first step the machine is loaded in each period with production gross requirements. Such a solution typically violates the capacity requirements. In the second step the production is sequenced from period T to 1. Whenever the capacity constraint is violated, we perform a procedure to eliminate overtime, which is the amount of the capacity violation. The third step is geared toward eliminating setups by anticipating productions, and the forth step tries to reduce holding costs by postponing productions. Finally, the last step attempts to find a better coordination between periods, eliminating, whenever possible, empty setups.

The heuristic starts with a lot-for-lot forward pass, allocating to each period the demand for that period ($X_{it} = d_{it}$, for every i, t), without considering the capacity constraints.

The second step works backwards and performs in each period both sequencing and amending procedures. In a given time period, the sequencing procedure is basically a *minmax* algorithm; in each pass a new arc (u, v) is selected due to either the variability of the input setup times of its initial node u (ΔI_u) or the variability of the output setup times of its terminal node v (ΔO_u). The variability is measured as the difference between the two smallest setup time values. The node with the maximum variability among the head and tail nodes is chosen. Then, the other node of the arc is the argument that minimizes the setup time of the selected arc. Since feasibility is placed over optimality in this step, the sequencing procedure tries to minimize the sequence dependent setup times. This procedure is detailed in Appendix A, together with an example. The option of considering the difference between the smallest and the second smallest setup time is preferred over the standard greedy approach of basing the selection only on the smallest setup time to "hedge" against the possibility of making a bad decision and later having very little room for correction. In addition, computational experiments have revealed this to be a superior criterion. For $t < T$, the last product to be scheduled is the first product in period $t + 1$ (the sink node is specified before generating the path in Algorithm 1).

Let $O_t = \max \left\{ 0, \sum_i p_i \cdot X_{it} + \sum_{ij} s_{ij} \cdot T_{ijt} - C_t \right\}$ denote the amount of overtime in period t. After the sequencing procedure, there can be time periods with positive overtime, i.e., with violated capacity. In the amending pass, we push overtime towards the previous period. While working on period t, we might create a new overtime at period $t - 1$. In case $O_t > 0$, in order to determine the product or set of products whose (partial or entire) lots will be moved backwards, the following

options are sequentially taken into account in the amending procedure (an option is only considered if the preceding one has not eliminated O_t).

(a) Let $S = \{i \in [N] : p_i \cdot X_{it} \geq O_t \text{ and } X_{i(t-1)} > 0\}$ and $k = \arg\min_{i \in S} h_i$.
Move O_t/p_k of X_{kt} to period $t-1$.

(b) Let $S = \{i \in [N] : 0 < p_i \cdot X_{it} < O_t \text{ and } X_{i(t-1)} > 0\}$, and let fs_{it} denote the minimum savings in setup time by not producing product i in period t. Formally,

$$fs_{it} = \begin{cases} \sum_j T_{ijt} \cdot s_{ij} & \text{if } \alpha_{it} = 1 \text{ and } \alpha_{i(t+1)} = 0 \text{ and } X_{it} > 0, \\ \min\limits_{\substack{X_{kt}>0, X_{jt}>0 \\ k \neq i, j \neq i}} [s_{ki} + s_{ij} - s_{kj}] & \text{if } \alpha_{it} = 0 \text{ and } \alpha_{i(t+1)} = 0 \text{ and } X_{it} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\alpha_{i(t+1)} = 1$ is not considered because it would modify the production sequence in downstream periods.
Let $k = \arg\min_{i \in S}(p_i \cdot X_{it} + fs_{it} : p_i \cdot X_{it} + fs_{it} \geq O_t)$. If k is not defined, then $k = \arg\max_{i \in S}(p_i \cdot X_{it} + fs_{it})$.
Move all X_{kt} to period $t-1$ and return to (a) if $O_t > 0$.

(c) Let $S = \{i \in [N] : p_i \cdot X_{it} \geq O_t\}$ and $k = \arg\min_{i \in S} h_i$.
Move O_t/p_k of X_{kt} to period $t-1$.

(d) Let $S = \{0 < p_i \cdot X_{it} < O_t\}$ and $k = \arg\min_{i \in S}(p_i \cdot X_{it} + fs_{it} : p_i \cdot X_{it} + fs_{it} \geq O_t)$.
If k is not defined, then $k = \arg\max_{i \in S}(p_i \cdot X_{it} + fs_{it})$.
Move all X_{kt} to period $t-1$, and return to (a) if $O_t > 0$.

We note that steps (c) and (d) are similar to (a) and (b), respectively, however, they lead to a higher capacity consumption in period $t-1$ since a new setup has to be performed. This is the reason for placing step (c) after step (b) as opposed to execute (a) and (c) simultaneously. Proceeding from the last period to the first period, the solution is either feasible or there is overtime in period one (and no feasible solution is generated). We point out that finding a feasible solution is NP-complete, Trigeiro et al. [1989]. If the first two steps do not find a feasible solution, then the entire heuristic fails.

The following steps try to improve the quality of the initial solution. Let $\phi = \{(i,t) : X_{it} > 0\}$, let fc_{it} denote the minimum savings in setup costs by not producing product i in period t (its expression is identical to fs_{it} with setup costs instead of setup times as one of the arguments), and let $Ca_t = \max\left\{0, C_t - \sum_i p_i \cdot X_{it} - \sum_{ij} s_{ij} \cdot T_{ijt}\right\}$ denote the available (idleness) capacity in period t. The third step of the heuristic is a backwards pass in time that seeks to avoid the cost and capacity consumption of a setup, by moving an entire lot of product i^* from period t to the closest previous period t_f^* in which product i^* is produced and there is enough idle capacity ($Ca_{t_f^*} > p_i \cdot X_{i^*t}$) to absorb the new lot, where

$$(i^*, t_f^*) = \arg\max_{\substack{(i,t_f) \in \phi, (i,t) \in \phi \\ t_f < t, Ca_{t_f} > p_i \cdot X_{it}}} [fc_{it} - h_i \cdot X_{it} \cdot (t - t_f)].$$

Note that these operations may destroy feasibility which needs to be recovered at the end.

The fourth step is a forward pass that seeks to reduce inventory holding costs, by shifting forward a fraction or an entire lot of product i from a source period t to a target period t_f in which product i is also produced ($X_{it_f} > 0$). This move is similar to the one presented by Hindi et al. [2003], but with a different purpose. Our objective is optimality, while their is feasibility. The candidate products have currently a positive inventory level ($I_{it} > 0$). In each iteration, the production quantity $Q^t_{i^*t^*_f}$ of product i^* is carried over from period t into period t^*_f, where

$$(i^*, t^*_f) = \underset{i\in[N], t_f\in\{1,\ldots,T-1\}}{\arg\max} \Big[h_i \cdot Q^t_{it_f} \cdot (t_f - t) : X_{it_f} > 0\Big],$$

$$Q^t_{it_f} = \min\{X_{it}, I_{it}, I_{i(t+1)}, \ldots, I_{i(t_f-1)}, C_{t_f}\}.$$

Finally, the last step looks for improvements in the links between adjacent periods, in a forward pass. The sequencing procedure of the second step may generate a schedule in which the machine is set up to produce product i at the beginning of period $t+1$ ($\alpha_{i(t+1)} = 1$), without producing i in that period ($X_{i(t+1)} = 0$). If there is a product j such that $X_{jt} > 0$, $X_{j(t+1)} > 0$, $\alpha_{j(t+1)} = 0, j \neq i$, then by assigning $\alpha_{j(t+1)} = 1$, $\alpha_{i(t+1)} = 0$ a setup is eliminated in period $t+1$. Let $S = \{j \in [N] : X_{jt} > 0,\ X_{j(t+1)} > 0,\ \alpha_{jt} = 0,\ \alpha_{j(t+2)} = 0\}$ be the set of candidate products. The last two conditions in the definition of S restrict the current analysis to periods t and $t+1$ (otherwise other periods would have to be rescheduled). To select product i^* from S we compute for every $j \in S$ variables ΔI_j and ΔO_j, in the same way as in the sequencing procedure, except that all quantities are computed with respect to c_{ij} (see Appendix A). To compute ΔO_j and ΔI_j we only take into account products that are produced in periods t and $t+1$, respectively. The selected product has the minimum aggregated variability $i^* = \underset{j\in S}{\arg\min}(\Delta O_j + \Delta I_j)$. Then, both periods t and $t+1$ are rescheduled with the input $\alpha_{i^*(t+1)} = 1$.

From all the steps described before, only the third and the fifth steps may generate unfeasible schedules due to the creation of overtime in a period. Whenever this occurs, we try to recover feasibility by applying the four options of step 2 from the last period with overtime backwards.

4 Computational Results

All computational experiments in this section were performed on an IBM personal computer with a 3.2 GHz CPU and 1GB of random access memory. CPLEX version 9.0 from ILOG was used as the mixed integer programming solver. We used the instance generator described in Section 2.1.1 to generate random instances. For each type of instance, characterized by the quadruple N, T, Cut and θ, we present the average of the values obtained in 10 randomly generated instances.

We first compare the LP relaxations of models F_1 and F_2, both strengthened with the (l, S) and (W_t) valid inequalities. Let F_{1LP} and F_{2LP} denote the values of the linear relaxations of F_1 and F_2, respectively, and let $F^\star_{1LP}$ and $F^\star_{2LP}$ denote the values of the linear relaxations of F_1 and F_2 including all the (l, S) and W_t cuts, respectively. The separation algorithms for inequalities (13) and (18) were coded in OPL version 3.7 from ILOG. We solved the minimum cut problem of the disconnected subtour separation algorithm with the Ford-Fulkerson's algorithm. The running times of the linear relaxation of F_2 strengthened with the (l, S) and (W_t) valid inequalities were always less than 150 seconds for all instances. As an example, with $T = 10$ and $N = 25$, model F_1 has 6760 columns and 4135 rows. Figure 5 presents the gap $\frac{F^\star_{2LP} - F^\star_{1LP}}{F^\star_{1LP}}$. As proved in Section 2.2, formulation F_2 is stronger than formulation F_1. The gap increases with the number of time periods.

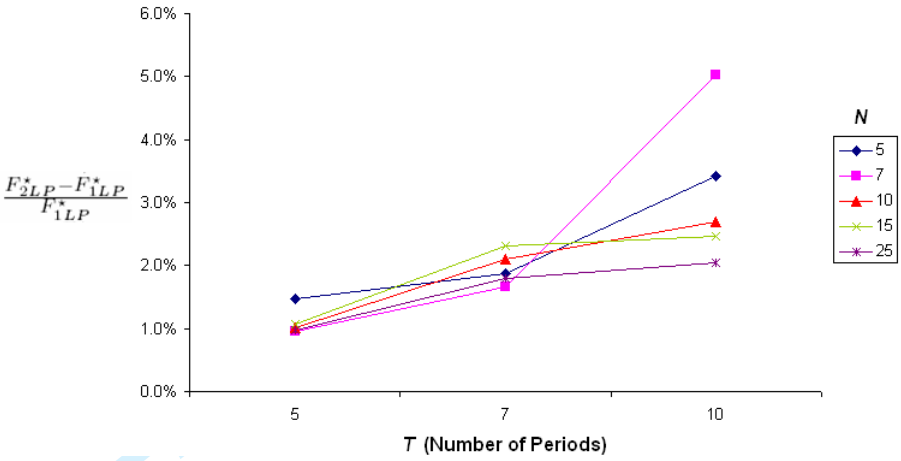


Figure 5: Comparison of $F^{\star}_{1LP}$ and $F^{\star}_{2LP}$ for $Cut = 0.6$ and $\theta = 100$

We next determine the impact of the (l, S) and (W_t) inequalities with respect to F_{2LP}. Let $F^{(l,S)}_{2LP}$ be the value of the linear relaxation of F_{2LP} including the (l, S) cuts and let $F^{(W_t)}_{2LP}$ be the value of the linear relaxation of F_{2LP} including the (W_t) cuts. The computational results are displayed in Table 4. Both (l, S) and (W_t) inequalities improve the lower bound, with an effect that increases (decreases) with the number of periods (products). Clearly, the improvement provided by the (l, S) cuts is especially noteworthy.

Table 5 reports the average deviation of the optimal solution from the lower bound $F^{\star}_{2LP}$, i.e., its integrality gap. As in Section 2.1.1, the empty buckets mean that at least one of the instances could not be solved optimally within a 3600 seconds time limit.

Table 4: Comparison (%) of $F^{(l,S)}_{2LP}$ and $F^{(W_t)}_{2LP}$ with F_{2LP} for $Cut = 0.6$ and $\theta = 100$

	$(F^{(l,S)}_{2LP} - F_{2LP})/F_{2LP}$			$(F^{(W_t)}_{2LP} - F_{2LP})/F_{2LP}$		
	T			T		
N	5	7	10	5	7	10
5	102.1	177.1	276.0	16.1	44.1	86.1
7	96.1	148.8	256.3	3.8	16.4	41.2
10	87.1	142.7	231.9	0.9	3.3	14.3
15	81.5	133.0	204.0	0.4	0.6	2.4
25	79.5	124.5	191.3	0.2	0.2	0.3

It is clear that the quality of the lower bound is significantly improved as the number of products (N) increases, and the quality seems to be independent of the number of periods (T). The results show that, on average, the integrality gap of the instances with $\theta = 50$ is lower than the integrality gap of the instances with $\theta = 100$. Nevertheless, this difference becomes smaller as the number of products in a problem instance increases. In both cases, from 25 onwards the gap is below 1%.

To generate an upper bound, we implemented the heuristic from Section 3 in C++ by using Visual Studio 6.0 as development environment. Tables 6 and 7 display the minimum, average, and

Table 5: Integrality gap (%) for $Cut = 0.6$

	$\theta = 50$			$\theta = 100$		
	T			T		
N	5	7	10	5	7	10
5	2.7	3.1	2.5	5.6	6.7	5.9
7	1.5	2.1	2.3	4.3	4.5	4.6
10	1.5	0.9	0.9	2.5	3.1	
15	1.0	0.9		1.5		
25	0.8	1.0		0.7		

maximum gap of the heuristic solution from the lower bound, for $\theta = 50$ and $\theta = 100$, respectively. The running times of the heuristic were always less than 0.4 seconds for all of the instances.

Table 6: Gap (%) between the lower and upper bounds for $Cut = 0.6$ and $\theta = 50$

	T		
N	5	7	10
5	2.8/6.4/12.7	2.5/8.3/19.2	2.0/6.4/15.6
7	2.3/6.0/9.9	2.7/7.0/12.7	2.6/6.6/10.3
10	5.6/9.5/18.0	4.7/8.9/13.4	3.3/7.7/14.9
15	4.7/9.7/14.6	4.9/10.0/18.9	6.3/10.1/12.4
25	6.5/9.9/11.9	6.0/11.1/14.9	8.6/12.0/16.3

minimum / average / maximum gap (%)

Table 7: Gap (%) between the lower and upper bounds for $Cut = 0.6$ and $\theta = 100$

	T		
N	5	7	10
5	6.8/15.4/22.6	10.3/15.8/18.5	5.1/15.2/26.4
7	4.6/12.9/19.0	6.4/13.5/19.1	9.5/14.5/17.9
10	6.6/13.6/20.1	9.6/13.4/15.5	7.7/13.5/20.7
15	12.9/17.0/20.5	10.4/16.7/21.4	11.9/16.7/23.1
25	20.3/24.0/26.4	21.9/24.1/25.9	20.2/23.6/26.7

minimum / average / maximum gap (%)

The heuristic found a feasible solution for all problem instances. The flexibility of the four options of step 2 of the heuristic enables us to find feasible solutions, even for tightly capacitated instances. The results indicate that for both $\theta = 50$ and $\theta = 100$ the heuristic performance deteriorates as the number of products increases. This situation is pronounced when the setup costs are considerable higher than the inventory costs ($\theta = 100$). The performance of the heuristic is not influenced by the number of periods. The amplitude of the gap between the lower and upper bounds tends to decrease as the number of product increases. Recall that in our production setting from the glass industry, typically $\theta = 50$ and thus Table 6 is more relevant. Based on Table 5, we also conclude that the most significant portion of the gap comes from the heuristic (and not from

a weak lower bound).

To the best of our knowledge, Gupta and Magnusson [2005] is the only paper in the literature to publish results (gaps between lower and upper bounds) for this problem. However, since inventories and production quantities are expressed in units of capacity (normalized to one), they adapted the parameters of the instance generator used by Haase and Kimms [2000]. By comparing our parameters with the parameters of their generator, their instances are similar to the case with $\theta = 50$. The authors present the performance of their heuristic from $N = 2$ to $N = 15$ for $T = 4$, and from $T = 2$ to $T = 15$ for $N = 4$. For problem instances with 15 products and 4 periods, Gupta and Magnusson [2005] report an average gap between lower and upper bounds of approximately 20%, and for problem instances with 4 products and 15 periods an average gap of approximately 30%. From Table 6 it clearly follows that our models and the heuristic are better.

5 Concluding Remarks

We have presented two novel exact formulations for modeling setup carryover in the challenging CLSP problem with sequence dependent setup times and costs. The models are simpler than other available in the literature. Note that all the models published so far only perform a setup whenever there is sufficient time available to do it entirely. In the case of very large setup times (as the ones observed in the glass container industry that may reach 120 hours), setup operations may start at the end of one period and finish at the beginning of the following period. This new feature has not yet been incorporated in CLSP models.

We have formally shown that one of the formulations is stronger than the other and we have implemented valid inequalities that enable us to reduce the integrality gap of the LP relaxation with a desirable property: the integrality gap reduces as the number of products increases (it is independent on the number of periods). It is clearly important to find strategies to combine even stronger valid inequalities based on the polyhedral structure of this problem with tighter reformulations. These inequalities should take into account the capacity constraints and the potential cycles and paths of products sequence in each time period.

We have described a five-step heuristic that is effective both in finding a feasible solution and in producing good solutions to these problems (measured by the gap between the lower and upper bounds). The simplicity of this heuristic enables its implementation as a building block of more complex heuristics and metaheuristics. It is worth pointing out that it may effectively generate good initial solutions and repair infeasibilities due to capacity requirements of solutions generated by any local search procedure.

Acknowledgments

The authors are grateful to Professor Stephen C. Graves from the Sloan School of Management for his helpful comments.

The first author is also grateful to the Portuguese Foundation for Science and Technology for awarding him the PhD grant SFRH/BD/23987/2005.

A The Sequencing Heuristic

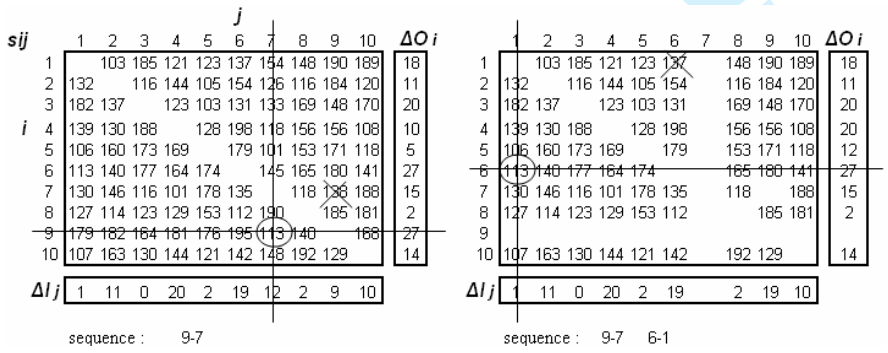
In this section, we present a heuristic to sequence N products in order to minimize the sequence dependent setup times s_{ij}. Since cycles are not taken into account and each product is produced,

the final sequence (path) has $N-1$ edges. The sequencing heuristic is given in Algorithm 1, where Θ is a large number, e.g., $\Theta = 1 + \max_{i,j} s_{ij}$.

loop $N-1$ times
  for $k = 1$ to N
    $i_1 = \arg\min_{i\in[N]}(s_{ik} : s_{ik} \neq \Theta)$
    $i_2 = \arg\min_{i\in[N]\setminus\{i_1\}}(s_{ik} : s_{ik} \neq \Theta)$
    $j_1 = \arg\min_{j\in[N]}(s_{kj} : s_{kj} \neq \Theta)$
    $j_2 = \arg\min_{j\in[N]\setminus\{j_1\}}(s_{kj} : s_{kj} \neq \Theta)$
    $\Delta O_k = s_{kj_2} - s_{kj_1}$
    $\Delta I_k = s_{i_2k} - s_{i_1k}$
  end for
  $i' = \arg\max_i(\Delta O_i), j'' = \arg\min_{j\in[N]}(s_{i'j})$
  $j' = \arg\max_j(\Delta I_j), i'' = \arg\min_{i\in[N]}(s_{ij'})$
  if $\Delta O_{i'} > \Delta I_{j'}$ or $(\Delta O_{i'} = \Delta I_{j'}$ and $s_{i'j''} \leq s_{i''j'})$ **then**
    $T_{i'j''t} = 1$
  else
    $T_{i''j't} = 1, i' = i'', j'' = j'$
  end if
  $s_{j'i''} = \Theta$
  for $k = 1$ to N
    $s_{i'k} = s_{kj''} = \Theta$
  end for
end loop

Algorithm 1: The Sequencing Procedure in Period t

Figure 6 gives an example of the above procedure. Instead of setting the time of already considered arcs to Θ, we delete such arcs.



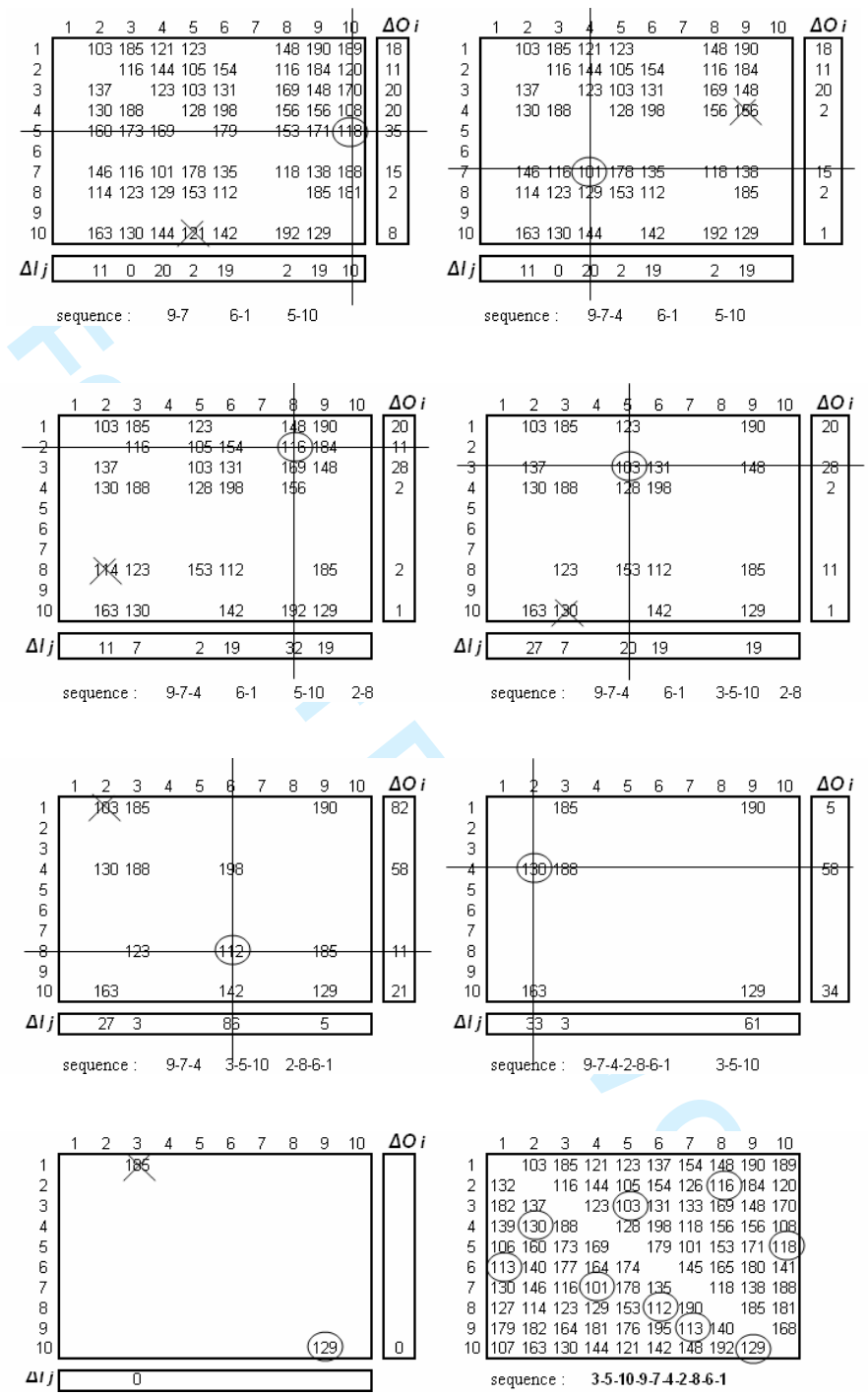


Figure 6: Illustrative example

References

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

B. Almada-Lobo, J. F. Oliveira, and M. A. Carravilla. A note on "The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times". *Computers and Operations Research*, doi: 10.1016/j.cor.2006.08.019, 2006.

I. Barany, T. J. Vanroy, and L. A. Wolsey. Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30(10):1255–1261, 1984.

A. R. Clark and S. J. Clark. Rolling-horizon lot-sizing when set-up times are sequence-dependent. *International Journal of Production Research*, 38(10):2287–2307, 2000.

A. Drexl and A. Kimms. Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research*, 99(2):221–235, 1997.

M. Gopalakrishnan. A modified framework for modelling set-up carryover in the capacitated lot-sizing problem. *International Journal of Production Research*, 38(14):3421–3424, 2000.

M. Gopalakrishnan, K. Ding, J. M. Bourjolly, and S. Mohan. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management Science*, 47(6):851–863, 2001.

M. Gopalakrishnan, D. M. Miller, and C. P. Schmidt. A framework for modeling setup carryover in the capacitated lot-sizing problem. *International Journal of Production Research*, 33(7):1973–1988, 1995.

D. Gupta and T. Magnusson. The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers and Operations Research*, 32(4):727–747, 2005.

K. Haase. *Lot sizing and scheduling for Production Planning*. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin, 1994.

K. Haase and A. Kimms. Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, 66(2): 159–169, 2000.

K. S. Hindi, K. Fleszar, and C. Charalambous. An effective heuristic for the CLSP with set-up times. *Journal of the Operational Research Society*, 54(5):490–498, 2003.

S. Kang, K. Malik, and L. J. Thomas. Lotsizing and scheduling on parallel machines with sequence-dependent setup costs. *Management Science*, 45(2):273–289, 1999.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

P. Porkka, A. P. J. Vepsalainen, and M. Kuula. Multiperiod production planning carrying over set-up time. *International Journal of Production Research*, 41(6):1133–1148, 2003.

C. R. Sox and Y. B. Gao. The capacitated lot sizing problem with setup carry-over. *IIE Transactions*, 31(2):173–181, 1999.

C. Suerie and H. Stadtler. The capacitated lot-sizing problem with linked lot sizes. *Management Science*, 49(8):1039–1054, 2003.

W. Trigeiro, L. J. Thomas, and J. McClain. Capacitated lot-sizing with setup times. *Management Science*, 35(3):353–366, 1989.