Pricing, relaxing and fixing under lotsizing and scheduling

Luis Guimarães[a,b], Diego Klabjan[b], Bernardo Almada-Lobo[a]
[a] INESC TEC, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
[b] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, USA
{guimaraes.luis@fe.up.pt, almada.lobo@fe.up.pt, d-klabjan@northwestern.edu}

We present a novel mathematical model and a mathematical programming based approach to deliver superior quality solutions for the single machine capacitated lotsizing and scheduling problem with sequence-dependent setup times and costs. The formulation explores the idea of scheduling products based on the selection of known production sequences. The model is the basis of a matheuristic, which embeds pricing principles within construction and improvement MIP-based heuristics. A partial exploration of distinct neighborhood structures avoids local entrapment and is conducted on a rule-based neighbor selection principle. We compare the performance of this approach to other heuristics proposed in the literature. The computational study carried out on different sets of benchmark instances shows the ability of the matheuristic to cope with several model extensions while maintaining a very effective search. Although the techniques described were developed in the context of the problem studied, the method is applicable to other lotsizing problems or even to problems outside this domain.

Key words: Lotsizing and scheduling; Sequence-dependent setups; Non-triangular setups; Column generation; MIP-based heuristics
History:

---

1. Introduction

In many production environments, production planning problems involve the determination of production lotsizes and sequence of different products on a single capacitated machine. Production lotsizes are driven by deterministic demand over the planning horizon. Switching between production runs of two different products triggers operations, such as machine adjustments and cleansing procedures, which consume scarce production time and can cause costs due, for example, to losses in materials. Under these conditions, production sequencing must explicitly take into account for these sequence–dependent setup times and costs. In this context, the need for simultaneous lotsizing and scheduling decisions arises. Production plans are created with the objective of minimizing

the overall costs consisting mainly of holding and setup costs, while satisfying the available capacity in each time period from which the expenditure in setup times is deducted. Examples of industries where these decisions must be taken concurrently are chemicals, drugs and pharmaceuticals, pulp and paper, textiles, foundries, glass container, and food and beverage, among many others (see Clark et al. [2011]).

Tackling real world problems requires to address special cases that may occur by introducing additional features into mathematical models. Among these realistic features are changeovers that do not respect the triangular inequality. When setups obey the triangular inequality with respect to both the setup time and costs, i.e. it is more efficient to change directly between two products than via a third product, at most one setup for each product per time period occurs. In some industries, contamination occurs when changing from one product to another implying additional cleansing operations. If a 'cleansing' or shortcut product can absorb contamination while being produced, replacing the cleansing operations, non-triangular setups appear. In their presence, models have to allow for more than one production run of each product per time period as it potentially reduces setup times and costs. Many examples of this type are known in the chemical, pharmaceutical, food and dyeing industries.

Mixed integer programming (MIP) models are unable to solve relevant size instances of the problem, suffering from its computational intractability (they are NP-hard by Bitran and Yanasse [1992]). State-of-the-art optimization engines either fail to generate feasible solutions to this problem or take a prohibitively large amount of computational time, as the computational experiments presented herein attest. Therefore, solving this class of problems requires the use of efficient solution approaches. Mathematical programming-based heuristics (Ball [2011]), also known as matheuristics (Maniezzo et al. [2010]), are algorithms which integrate exact and heuristic search techniques. Exact algorithms provably achieve optimal or quasi-optimal solutions, yet the size of tractable problems is limited. On the other hand, metaheuristics (heuristic search) are tailored to solve large-scale combinatorial optimization problems exploring large size neighborhoods efficiently. The underlaying idea of matheuritics is to seek the best trade-off between the effectiveness of exact approaches and the efficacy of metaheuristics. Furthermore, in general, these algorithms are flexible enough to cope with different model extensions and new features.

The motivation for this work is the development of a flexible solution methodology integrating exact and approximate methods able to solve lotsizing and scheduling problems of relevant sizes and features present in real world applications. We introduce a new MIP model for the single machine capacitated lotsizing and scheduling problem (CLSD) that accommodates non-triangular set-

tings. The model schedules production based on the selection of feasible production sequences. We develop a pricing heuristic (*SeqSearch*) to generate the sequences to be incorporated in the model since including all possible sequences is intractable as its number grows exponentially with the number of products. To obtain superior quality solutions to the CLSD, we develop a construction and improvement heuristics combining *SeqSearch* with mathematical programming-based heuristics. The construction heuristic (*Relax-Price-Fix*) uses a rolling horizon approach to sequentially construct a solution to the problem, while the improvement heuristic (*Fix-Price-Optimize*) attempts to partially optimize a feasible solution by solving small MIP subproblems. Different neighborhood structures are explored during the local search to avoid local entrapment. The two driving principles of neighborhood structures definition are to consider subproblems having a small number of consecutive times periods with all products or a small set of products over a larger portion of the planning horizon.

Our contributions are as follows. To the best of our knowledge, the new MIP model is the first one to capture non-triangular settings based on the selection of a single sequence in each time period. This is a non-trivial extension since products can repeat. An important ingredient of our solution methodology is a formulation that combines a compact and an extended formulation within a single model. This formulation trades-off accuracy and computational complexity. On the algorithmic front, we create a new MIP-based construction heuristic using this hybrid formulation. Another very important contribution concerns our novel ideas to use column generation for local search within lotsizing problems. The methodology exposed in this paper can be generalized to different lotsizing problems or even to problems outside this research field.

The remainder of this paper has the following structure. Section 2 presents the new formulation for the CLSD. Section 3 describes our solution approach to solve the CLSD and its main building blocks. A series of computational experiments with different problem sets having distinct features are shown in Section 4. Finally, Section 5 is devoted to final remarks, conclusions from this work and some future research directions are pinpointed. We conclude the introduction with an overview of the most relevant work.

1.1 Literature review

The field of lotsizing and scheduling has received an increased attention from the research community due to its inherit applicability to real world problems as shown in the reviews by Drexl and Kimms [1997], Zhu and Wilhelm [2006], Jans and Degraeve [2008] and, recently, by the special issue Clark et al. [2011]. This applicability can only be achieved with adequate solution approaches,

most of which are based on mathematical representations of the problem. Mathematical formulations for lotsizing and scheduling assume a planning horizon divided into a finite number of time buckets. These discrete time formulations can be grouped into two types: large and small bucket models.

Large bucket models allow for more than one setup per time period. Sequencing decisions within each time period use decision variables similar to those of routing problems formulations and require sub-tour elimination constraints to correctly represent production sequences. Almada-Lobo et al. [2007] present an exact formulation for the CLSD when setups obey the triangular inequality, which was extended by Menezes et al. [2011] to the non-triangular case using an exponential number of constraints. Sarin et al. [2011] present a formulation with a polynomial number of sub-tours elimination constraints through multi-commodity-flow-type constraints. All these works deal with compact formulations, while we develop an extended formulation.

On the other hand, in small bucket models the production sequence comes for free directly from the assumption of allowing at most one setup per period. These models do not impose any restriction on the setup configuration and neither require sub-tour elimination constraints. The general lotsizing and scheduling problem (GLSP) model described by Fleischmann and Meyr [1997] and Meyr [2000] is the most flexible of such models. In the GLSP, time periods are divided into micro-periods using an *a priori* defined parameter. The number of micro-periods may account for the maximum number of setup operations allowed in each period, or divide each time period (e.g. weeks) into many shorter periods (e.g. days, hours or shifts). Hence, the model size is dramatically increased and/or multiple optimization runs with different parameter choices must the conducted to achieve optimality. Furthermore, Wolsey [2002] shows that the linear relaxation of small bucket models results in much weaker lower bounds in comparison to large bucket models.

The aforementioned models can be called compact or product related formulations, as sequencing decisions are taken from decision variables indexed by product. An alternative model may select the production sequence from a set of available production sequences, which are acceptable in each time period. We call these models extended or sequence related formulations. Examples are given in Haase and Kimms [2000] and Kovács et al. [2009] for big bucket formulations, and Kang et al. [1999] for a small bucket model. Sequence related formulations usually result in simpler models as sub-tour elimination constraints and auxiliary decision variables used to ordinate products are not required. However, as the number of products increases, the number of sequences grows exponentially. The mathematical formulation presented in this paper is, to the best of our knowledge, the first sequence related formulation considering a large bucket model for

non-triangular setups.

Most solution procedures for the CLSD combine heuristics with exact methods. In Meyr [2000] the small bucket mathematical model is solved by embedding a dual network flow algorithm into threshold accepting and simulated annealing. These procedures were later extended for the case of parallel machines in Meyr [2002].

With the main purpose of solving specific instances Kang et al. [1999] present a branch-and-price algorithm for a small bucket sequence related formulation of the CLSD. It consists in dividing the entire production schedule into smaller production sequences, which the authors call split-sequences. For each period t the production sequence is composed of L_t split-sequences, resembling subperiods in product related formulations. To address the large number of split-sequences arising they propose a column generation based heuristic, where in each iteration the new split-sequences are obtained by an enumeration algorithm with an additional parameter *maxBR*, the maximum number of products in the split-sequence. To generate upper bounds, two different algorithms apply mathematical programming methods heuristically: one truncates the branch-and-bound search with respect to the number of fractional variables, while the other iteratively executes local search to improve the incumbent solution. A major disadvantage of this methodology is that to solve a given problem multiple runs are needed with different values of L_t and *maxBR*.

Progressive interval heuristics (Federgruen et al. [2007]) are MIP-based heuristics which solve a series of partially relaxed MIP subproblems to construct an initial feasible solution to the original MIP. Relax-and-fix heuristic starts from the first period in the planning horizon and progressively moves forward fixing the setup variables at their optimal value obtained in previous iterations. It is applied in Ferreira et al. [2009] to a practical case in the beverage industry on a small bucket model. Similarly to relax-and-fix, the 'exchange' improvement heuristic of Pochet and Wolsey [2006] and the fix-and-optimize version of Sahling et al. [2009] decompose the set of integer variables in the original MIP to create MIP subproblems to re-optimize. At each iteration, integer setup variables are fixed to their previous best value, apart from a small subset in which they are required to take any integer value, defining the subproblem to be optimized. Based on a large bucket model for the CLSD with triangular setups, James and Almada-Lobo [2011] integrated fix-and-optimize in a stochastic local search algorithm to improve the initial solution obtained with the relax-and-fix heuristic, delivering solutions within a small deviation from theoretical lower bounds.

Our solution approach, based on a large bucket sequence related model, integrates column generation in relax-and-fix and fix-and-optimize schemes. The pricing heuristic developed, which deals with the exponential number of production sequences, replaces the application of a branch-

and-price algorithm (Barnhart et al. [1996]) as the solution of the linear relaxation of our model is likely to be fractional. Moreover, as in Muter et al. [2010] and Alvelos et al. [2010], the purpose of column generation is to provide good partial solutions that can latter be combined by solving the master model or via a metaheuristic. In our case, integer solutions are found by applying relax-and-fix to the master model and they are improved by using fix-and-optimize, which is a clear difference to the work of Kang et al. [1999] where column generation is embedded in the branch-and-bound tree to generate new integer solutions. Following the classification of Blum et al. [2011] our approach falls into the categories of the hybridization of metaheuristics with tree search techniques and problem relaxation.

2. New models for the CLSD with non–triangular setups

2.1 A sequence related model

In this section we introduce a new sequence related model for the CLSD with non-triangular setups. This model constitutes the basis of our heuristic procedures. Throughout the exposition, let us consider set $\mathscr{N}$ composed of N products indexed by $i, j = 1, \ldots, N$ to be produced on a single capacitated machine over a finite planning horizon of T periods, defining a set $\mathscr{T}$ indexed by $t, l = 1, \ldots, T$. The following data is associated with this problem:

d_{it}	demand of product i in period t (units),
h_i	holding cost of one stock unit of product i (cost/unit),
cap_t	capacity of the machine in period t (time),
p_i	processing time of product i (time/unit),
b_{it}	upper bound on the production quantity of product i in period t (units),
sc_{ij}	cost incurred to set up the machine from product i to product j (cost),
st_{ij}	time needed to set up the machine from product i to product j (time).

The mathematical model stated next is a big bucket sequence related model. The setup state is carried over among adjacent periods, i.e. the setup state is preserved even over idle time. Moreover, setup crossovers are not allowed, which force setup operations to be performed within the time period, without spanning to the following period. The validity of this assumption relies on the fact that we are dealing with a big-bucket model. As several products can be produced per period (e.g. week), interesting production plans should not be excluded. Stockouts are not accepted, which is a common setting in deterministic demand environments, and no initial inventory is considered.

However, such extensions are relatively straightforward. Finally, more than one setup may be performed to each product within a time period to address instances where setups do not obey the triangular inequality.

The first set of decision variables captures lotsizing decisions. To this end, let variables X_{itl} define the quantity of product i produced in period t to satisfy demand in period l. A model using such variables is usually referred to as a facility location model (FLM), originally proposed by Krarup and Bilde [1977] for the single-item problem. The FLM is known to be strong for lotsizing problems, giving tight lower bounds. To determine scheduling decisions, let $\mathscr{S}_t$ denote the set of all S_t feasible production sequences to schedule products on the machine in period t, indexed by $s = 1, \ldots, S_t$. Associated with each sequence we define the following parameters:

$\widehat{sc}_s$	setup cost incurred if sequence s is selected,
$\widehat{st}_s$	setup time incurred if sequence s is selected,
f_{is}	(=1) if product i is first in sequence s,
l_{is}	(=1) if product i is last in sequence s,
e_{is}	(=1) if the machine is ever set up for product i in sequence s,
a_{is}	number of setups performed to product i in sequence s.

Product sequencing can be modeled by the following decision variables:

W_{ts}	(=1) if sequence s is chosen in period t,
U_{it}	(=1) if at least one setup is performed to product i in period t,
Y_{it}	number of setups performed to product i in period t,
Z_{it}	(=1) if the machine is set up for product i at the beginning of period t.

Our sequence-related MIP model for the CLSD reads:

$$(FS) \quad \text{Min} \quad \sum_{t\in\mathscr{T}}\sum_{s\in\mathscr{S}_t} \widehat{sc}_s \cdot W_{ts} + \sum_{t\in\mathscr{T}}\sum_{\substack{l\in\mathscr{T}\\ l>t}}\sum_{i\in\mathscr{N}} (l-t)\cdot h_i \cdot X_{itl} \tag{1}$$

subject to:

$$\sum_{\substack{t\in\mathscr{T}\\ t\le l}} X_{itl} = d_{il} \qquad \forall i\in\mathscr{N}, l\in\mathscr{T} \tag{2}$$

$$\sum_{i\in\mathscr{N}}\sum_{\substack{l\in\mathscr{T}\\ l\ge t}} p_i \cdot X_{itl} + \sum_{s\in\mathscr{S}_t} \widehat{st}_s \cdot W_{ts} \le cap_t \qquad \forall t\in\mathscr{T} \qquad (\lambda_t) \tag{3}$$

$$X_{itl} - d_{il}\cdot(U_{it}+Z_{it}) \le 0 \qquad \forall i\in\mathscr{N}, t, l\in\mathscr{T}, l\ge t \tag{4}$$

$$\sum_{s\in\mathscr{S}_t} f_{is}\cdot W_{ts} = Z_{it} \qquad \forall i\in\mathscr{N}, t\in\mathscr{T} \qquad (\theta^f_{it}) \tag{5}$$

$$\sum_{s\in\mathscr{S}_t} l_{is}\cdot W_{ts} = Z_{i,t+1} \qquad \forall i\in\mathscr{N}, t\in\mathscr{T} \qquad (\theta^l_{it}) \tag{6}$$

$$\sum_{i\in\mathscr{N}} Z_{it} = 1 \qquad \forall t\in\mathscr{T} \tag{7}$$

$$\sum_{s\in\mathscr{S}_t} e_{is}\cdot W_{ts} = U_{it} \qquad \forall i\in\mathscr{N}, t\in\mathscr{T} \qquad (\alpha_{it}) \tag{8}$$

$$\sum_{s\in\mathscr{S}_t} a_{is}\cdot W_{ts} = Y_{it} \qquad \forall i\in\mathscr{N}, t\in\mathscr{T} \qquad (\pi_{it}) \tag{9}$$

$$(X_{itl}, W_{ts}) \ge 0, \quad (U_{it}, Z_{it}) \in \{0,1\}, \quad Y_{it}\in\mathbb{N} \quad \forall i\in\mathscr{N}, t, l\in\mathscr{T}, s\in\mathscr{S}_t. \tag{10}$$

The objective function (1) minimizes the total holding and setup costs. Demand fulfillment is expressed in constraints (2). Constraints (3) guarantee that the total production and setup times in each period do not exceed available capacity. Requirements (4) ensure for each period that a product is only produced in case the machine is properly configurated. Such a configuration might have been carried over from the previous period or resulted from a setup in that period. Constraints (5) and (6) link the machine initial configuration in each period with the first and last product in the selected sequence, implying that if a given product is the first of the sequence in the current period, then it has to be the last in previous period (setup carry-over). Constraints (7) state that the machine is set up for exactly one product at the beginning of each time period. Product setup decisions are linked with sequence selection through constraints (8) and (9). Variable domains are defined in (10). The model extension to capture minimum and maximum lot sizes is shown in Appendix A.

Remark 1. *Any model capable of tackling non-triangular setups can also address triangular setups. In the presence of setups that obey the triangular inequality, decision variables Y_{it} and constraints* (9) *are not required. In fact, when the triangular inequality holds, any optimal solution for the CLSD contains at most one setup for each product in each time period, turning Y_{it} redundant in the presence of U_{it}. Nevertheless, the consideration of maximum lot sizes may result into more*

than one production run in an optimal solution, requiring once again variables Y_{it} and constraints (9).

Remark 2. *Integrality of variables W_{st} is relaxed as the integrality of variables U_{it}, Y_{it}, Z_{it}, constraints* (5)-(9) *and the minimization of the setup cost imply the selection of a single sequence in the pool.*

2.2 A mixed product and sequence related model

The sequence related model FS just presented has an exponential number of variables W_{st} making a full implementation impracticable. Therefore, to achieve an efficient model implementation sequence assignment variables W_{st} have to be dynamically generated. The utilization of a column generation algorithm to provide the required variables introduces an additional computational effort which can compromise the efficiency of model FS. We introduce a hybrid formulation combining the sequence related formulation and a product related formulation, listed in Appendix B, to relief the effort spent in generating new columns.

Consider a partition of the set of planning periods $\mathscr{T}$ into two disjoint subsets $\mathscr{T}_s$ and $\mathscr{T}_p$. Model FS is applied to subset $\mathscr{T}_s$ and sequencing decisions are obtained through variables Y_{it}, U_{it}, Z_{it} and W_{st}. In the remaining portion of the planning horizon, subset $\mathscr{T}_p$, the product related model of Appendix B determines production sequences using variables T_{ijt}, G_{it} and Z_{it}. Note that Z_{it} ensure the proper linking between the two formulations as they appear in both. We omit the hybrid formulation since it is a straightforward combination of the two formulations.

Despite this effort to manage computational intractability, large-scale mathematical models arising in real-world problems still require additional measures. We have developed a solution approach to the CLSD with non-triangular setups based on the two described models. The next section describes the proposed heuristic in which mathematical programming techniques are combined with metaheuristics, aiming to achieve a flexible method able to tackle different features of this problem, while delivering superior quality solutions.

3. Solution approach

This section is devoted to our solution approach to solve the CLSD, which we call *Price-and-MIP* (*P&MIP*). The method is composed of three main building blocks, as depicted in Figure 1.

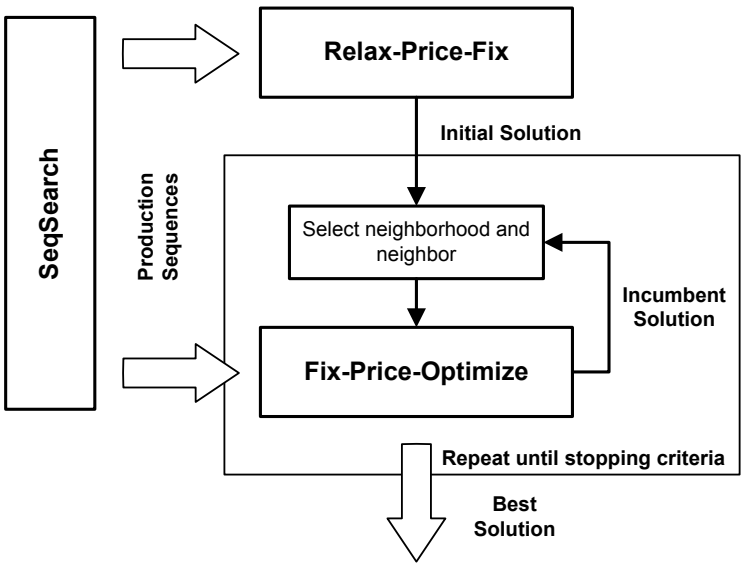

Figure 1: *P&MIP* Flowchart

SeqSearch A pricing heuristic which deals with the large number of variables W_{st} present in our model formulation. It identifies a subset of production sequences to be kept in the model at each step of the solution approach.

Relax–Price–Fix An MIP-based construction heuristic to build an initial feasible integer solution to the CLSD. It essentially results from combining the relax–and–fix framework with *SeqSearch*.

Fix–Price–Optimize An improvement heuristic which attempts to improve a feasible solution by decomposing the original MIP problem into smaller subproblems to be solved. It also combines mathematical programming and *SeqSearch*.

SeqSearch is embedded into the construction and improvement heuristics. It is responsible to generate, update and manage the pool of sequences preserved in FS. The overview of the various stages of the approach is given in Figure 2. A feasible initial solution to the CLSD is obtained through *Relax-Price-Fix* by progressively fixing integer variables in model FS in a rolling horizon fashion. The construction heuristic is described in more detail in Section 3.2. To improve the incumbent feasible integer solution we use *Fix-Price-Optimize* (see Section 3.3), which re-optimizes parts of a feasible solution. As shown in Figure 2 we rely on a systematic exploration of different neighborhoods to escape from local entrapment when applying the improvement heuristic.

In the following subsections we detail the main features of these building blocks.

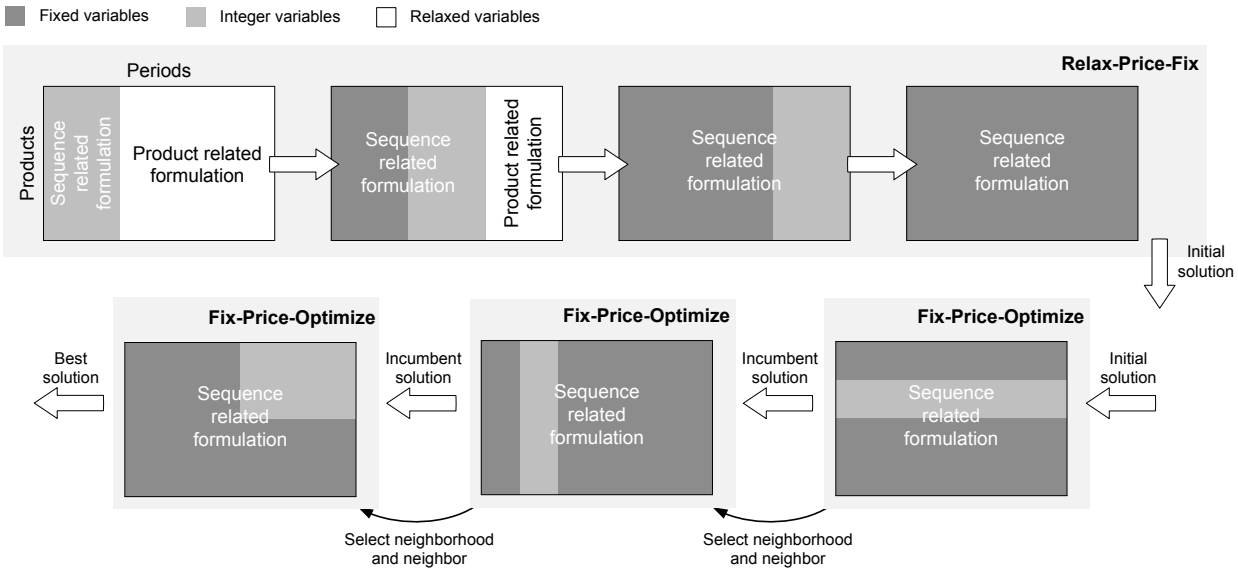

Figure 2: *P&MIP* overview

3.1 *SeqSearch*: Pricing production sequences

The purpose of *SeqSearch* is to identify the set of production sequences (related to variables W_{st}) to include in model FS and iteratively finding an integer solution. In Figure 3 the outline of the heuristic is presented. The overall procedure is composed of two nested loops, an inner and an outer loop. In the inner loop a column generation algorithm manages and updates sets of period production sequence. The outer loop guides the search of production sequences towards integer solutions. It corresponds to an LP-driven diving heuristic (see Pochet and Wolsey [2006]), which is in fact a way to perform a depth-first search strategy in the branch-and-bound tree. Iteratively, the information from the incumbent LP solution is used to fix integer variables to an integer value, until all variables are fixed (or the problem becomes infeasible).

The procedure starts with the definition of model FS using an initial subset $\mathscr{S}_t^{'}$ of feasible production schedules in each period $\mathscr{S}_t^{'} \subset \mathscr{S}_t$ (restricted problem - FS_r). In each iteration of the inner loop the linear relaxation of FS_r is solved and the dual information obtained is used to update and manage the sequence pool in each period. The pool is trimmed if the maximum number of sequences is exceeded. Proving optimality of the relaxed FS_r can imply a strong computational effort due to the tailing-off effect presented by the column generation method and the difficulty of the subproblems to solve. Therefore, a lower bound is calculated based on the reduced costs to

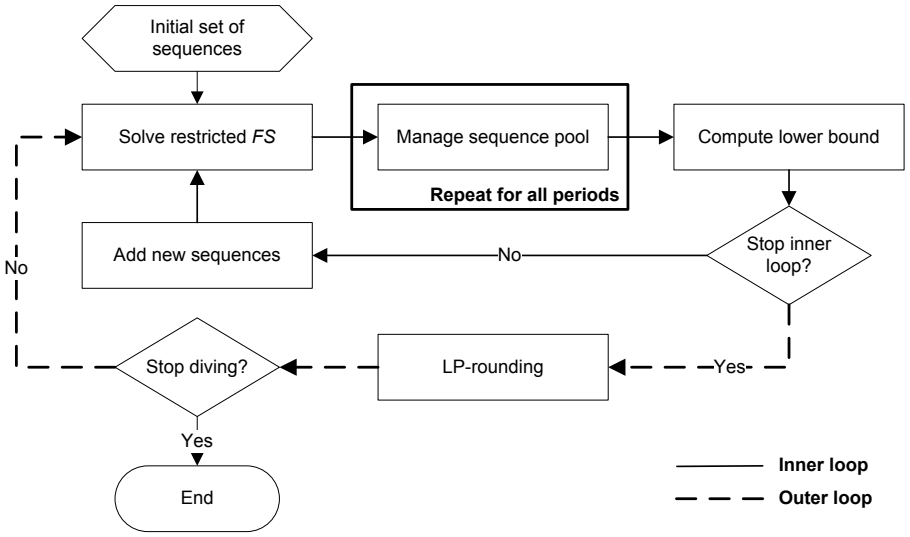

Figure 3: Outline of *SeqSearch*

invoke an early termination of the inner loop.[1] The loop is stopped if the percentage difference between the upper bound provided by the current solution of FS_r and the lower bound is less than a predefined threshold. Other stopping criterion for the inner loop can be: (1) no more negative reduced cost sequences, (2) iteration limit and (3) time limit.

The objective of generating 'good' production sequences used to obtain superior quality integer solutions for the CLSD may not be achieved only by the inner loop. The column generation algorithm is mainly concerned in solving the linear relaxation of FS_r. Therefore, although some of the production sequences generated may contribute to the final purpose of the heuristic, other may only be useful to find the LP-optimum. Hence, after the termination of the inner loop, the outer loop obtains a primal solution. By rounding integer variables of FS, it guides the inner loop to generate sequences useful for integer solutions. The diving scheme rounds the least fractional variables hierarchically first on set U, then on set Y and finally on set Z.

The search for production sequences ends when a feasible integer solution is found or the model becomes infeasible after fixing some of the integer variables. The final output of the heuristic is an updated pool of production sequences for the time periods considered together with an integer solution if one is found.

Next we present in more detail the subproblem solved to generate new production sequences.

[1] Let $z(FS_r^k)$ be the objective value of FS_r at iteration k of our column generation algorithm, rc_t^k be the minimum reduced cost associated with the solution of the pricing subproblem in time period t and $z(FS)$ the optimal value of the LP relaxation of FS. A lower bound on $z(FS)$ can be calculated by the following expression: $z(FS_r^k) \geq z(FS) \geq z(FS_r^k) + \sum_{t \in \mathcal{T}} rc_t^k$

3.1.1 Subproblems

Consider λ_t, θ^f_{it}, θ^l_{it}, α_{it} and π_{it} to be the dual variables associated with constraints (3), (5), (6), (8) and (9), respectively. For a specific time period t the subproblem objective function, which represents the reduced cost associated with variable W_{ts}, becomes:

$$(sub_t) \quad \underset{\chi_{ij}}{\text{Min}} \quad \widehat{sc}_s - \widehat{st}_s \cdot \lambda_t - \sum_{i \in \mathcal{N}} \left(f_{is} \cdot \theta^f_{it} - l_{is} \cdot \theta^l_{it} - e_{is} \cdot \alpha_{it} - a_{is} \cdot \pi_{it} \right). \tag{11}$$

The subproblem arising in each time period is a generalization of the prize collecting traveling salesman problem introduced by Balas [1989] as nodes can be visited more than once. Network $G = (V, A)$ consists of node set $V = \mathcal{N} \cup \{0, N+1\}$ and arc set A. Node 0 is the source and node $N+1$ the sink while the remaining nodes represent products (see Figure 4). The source and the sink are used to identify the starting and ending products of the production sequence, hence an arc connecting the source to a product means a carry over from the previous period and, similarly, an arc connecting a product to the sink represents a carry over to the next period. There is a prize ρ_i for visiting node i, as well as travel costs c_{ij} for traversing arcs (i, j). A node is considered to be visited if it follows another node other than the source. In addition, no penalties are considered for excluding nodes from the walk. The objective is to find the minimum cost walk through the network from the source to the sink. To mathematically state the subproblem, we introduce integer

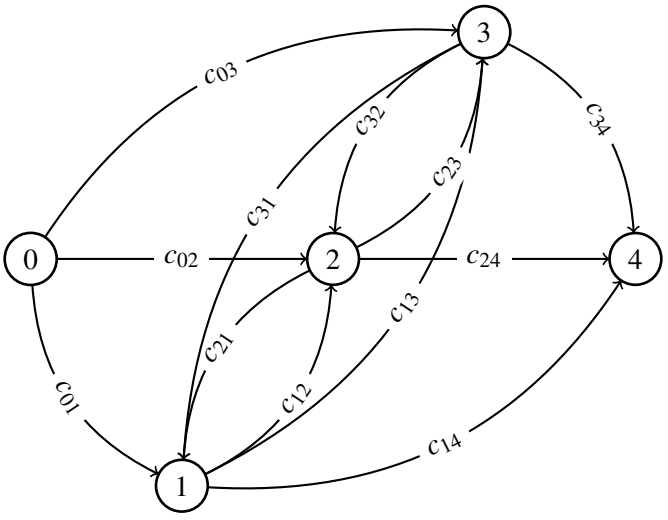

Figure 4: Network of the subproblem for $N = 3$

decision variables χ_{ij} representing the number of times arc (i, j) is traversed and y_i which equals 1 if node i is visited at least once or 0 otherwise. The MIP model formulation for the subproblem is presented in Appendix C.

3.2 *Relax–Price–Fix*: Constructing an initial solution

To create a feasible integer solution to the CLSD, we have developed a construction heuristic based on the relax-and-fix scheme (Pochet and Wolsey [2006]) and the formulations discussed in Section 2. Integer variables of the original MIP problem are partitioned into subsets. Then by sequentially solving a collection of partially relaxed MIP subproblems an integer solution is found to the original MIP. At each iteration of the heuristic, integer variables can be grouped into three different subsets: (1) variables whose values have been fixed in previous iterations, (2) variables required to be integer in the current stage and (3) relaxed variables. As the heuristic progresses these three subsets are being updated. The heuristic finishes when a feasible integer solution is found to the entire problem, or when a subproblem results infeasible. The partitioning strategy of the integer variables of the original MIP determines both the solution quality and computational effort. The larger the subsets, the better the solution quality, however a more complex MIP subproblem has to be solved in each iteration.

Our strategy relies on time partitioning of the original MIP, where the planning horizon is divided into time intervals containing a subset of in-time adjacent time periods. This partition creates a rolling horizon approach, as the heuristic starts by solving subproblems corresponding to the first time periods and progressively moves towards the end of the planning horizon. Let k be the current relax–and–fix heuristic iteration and let t_s^k and t_f^k denote the starting and ending periods of the current subset. The subproblem to be solved in iteration k, labeled as $subMIP^k$, corresponds to the model FS where equations (10) are replaced by:

$$(X_{itl}, W_{ts}) \geq 0 \qquad \forall i \in \mathscr{N}, t, l \in \mathscr{T}, s \in \mathscr{S}_t \tag{12}$$

$$U_{it} = \overline{U}_{it}, Z_{it} = \overline{Z}_{it}, Y_{it} = \overline{Y}_{it} \quad \forall i \in \mathscr{N}, t \in \mathscr{T}, t < t_s^k \tag{13}$$

$$(U_{it}, Z_{it}) \in \{0,1\}, \quad Y_{it} \in \mathbb{N} \quad \forall i \in \mathscr{N}, t \in \mathscr{T}, t_s^k \leq t \leq t_f^k \tag{14}$$

$$(U_{it}, Z_{it}, Y_{it}) \geq 0 \qquad \forall i \in \mathscr{N}, t \in \mathscr{T}, t > t_f^k. \tag{15}$$

Figure 5 depicts two successive iterations of the heuristic. Time periods colored in dark gray are those in which the value of integer variables are fixed to the solution obtained in previous iterations (equations (13)). The subset of integer variables belonging to period t_s^k up to period t_f^k (periods in light gray) are restricted to assume integer values (equations (14)). Finally, the integer variables of later periods (filled in white) are relaxed to take rational values (equations (15)). Consider σ to be the number of time periods in the subset of in-time adjacent periods and β to be the number of

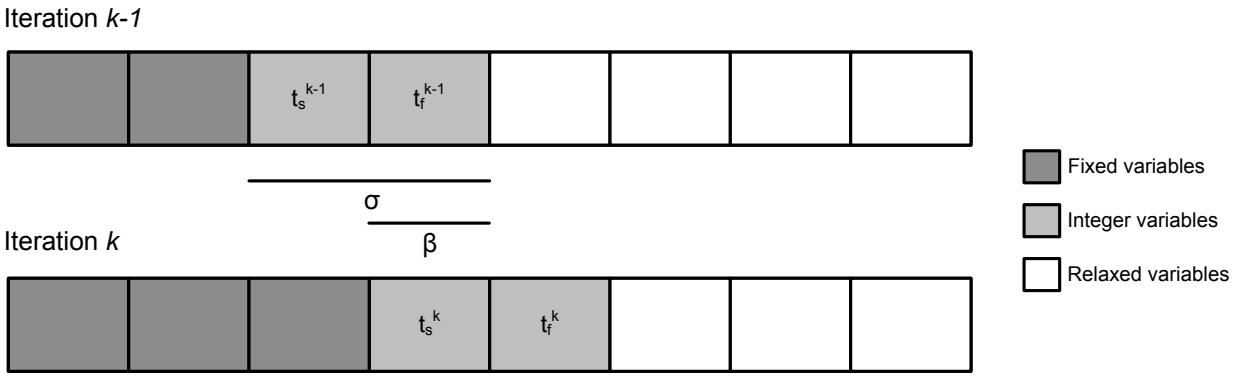

Figure 5: Successive iterations of *Relax–Price–Fix*

overlapping time periods between iterations. At the end of each iteration, integer variables from period t_s^k up to period $t_s^k + \sigma - \beta - 1$ are fixed to their respective value in the solution obtained by solving $subMIP^k$. The heuristic proceeds by moving t_s^k and t_f^k, according to $t_s^k = t_f^{k-1} - \beta + 1$ and $t_f^k = \min\{t_f^{k-1} + \sigma - \beta, T\}$, where σ is the number of time periods in each time partition and β is the number of time periods overlapping between iterations (in Figure 5 we have $\sigma = 2$ and $\beta = 1$).

Due to the nature of the original MIP model used to run the relax-and-fix heuristic (exponential number of decision variables of type W_{st}), *SeqSearch* was embedded within the relax-and-fix framework. Furthermore, aiming for a more efficient method, during *Relax–Price–Fix* the hybrid model discussed in Section 2.2 is used. Time periods spanning from the beginning of the planning horizon up to t_f^k define $\mathscr{T}_s$ and scheduling decisions are made using model FS. For later time periods ($t \in \mathscr{T}_p, t > t_f^k$) the product related formulation provides a relaxed solution in order to estimate future costs of the schedule. In each iteration, we first solve the linear relaxation of $subMIP^k$ using *SeqSearch* identifying new production sequences to add to model FS in time periods colored in light gray (*SeqSearch* is called for each time period $t \in [t_s^k, t_f^k]$), considering their respective dual values. Therefore, new production sequences are only generated for periods requiring integrality for integer variables as in previous periods these decisions have already been fixed and in later periods sequences are estimated by the product related model. Restricting the generation of new sequences to a low number of time periods in each iteration relieves the computational burden of the construction heuristic. Between two consecutive iterations of the construction heuristic, the hybrid model is update converting the product related formulation into the sequence related formulation for periods $t \in [t_f^{k-1} + 1, t_f^k]$.

3.3 *Fix–Price–Optimize*: Improving solution quality

Let $\mathscr{T}^{'}$ define a subset of periods and $\mathscr{N}^{'}$ a subset of products. The subproblem aiming to improve the current best solution corresponds to fixing the integer variables not present in these two sets to their incumbent value so that changes to the value of the integer variables are only allowed within the defined subsets. Before solving the subproblem, *SeqSearch* heuristic is performed to identify new production sequences to add into the sequence pool of model FS for the subset of periods $\mathscr{T}^{'}$ to be re-optimized, based on the dual values of the variables related to the products and periods in the defined subsets. Naturally, new production sequences are created taking into account the setups that will remain unchanged.

We systematically explore changes in $\mathscr{N}^{'}$ and $\mathscr{T}^{'}$ in order to avoid local minima. Consider an ordered finite set of user-defined neighborhood structures N_n, $(n = 1, \ldots, n_{max})$, where n denotes the nth neighborhood structure. Each neighborhood structure contains several neighbors. After solving a subproblem from the current neighborhood structure the new solution objective value is compared with the previous best solution value. In case of an improvement, the search restarts at the first neighborhood structure ($n = 1$). Otherwise, the number of failed attempts within the current neighborhood structure is increased. We allow a limited number of failures before switching to the next neighborhood structure in the ordered set.

Neighborhood structures are defined by the number of products $N^{'}$ and the number of adjacent periods $T^{'}$ to be re-optimized. A neighbor corresponds to the selection of $\mathscr{N}^{'} \subseteq \mathscr{N}$ of cardinality $N^{'}$ and $\mathscr{T}^{'} \subseteq \mathscr{T}$ of cardinality $T^{'}$, defining the set of 'released' variables and the MIP subproblem to solve. Hence, neighborhoods contain all possible combinations of $\mathscr{N}^{'}$ and $\mathscr{T}^{'}$ of given cardinalities. Since our neighbor evaluation is a computational expensive process a full evaluation of the neighborhoods is unpractical. Therefore, a stochastic process controls neighbor selection to conduct a partial neighborhood search.

When starting the exploration of a given neighborhood structure, scores τ_i and ω_t are assigned to each product and period, respectively. Initially, at the beginning of a neighborhood phase, we set all of them to 1. As products and periods are selected their score is updated so that the more frequent (number of times selected during the neighborhood exploration) and recently (number of neighbors explored since last selected) a given product or period has been selected, the lower is its score (weighted average of both criteria). The neighbors scoring method used is similar to the one described in James and Almada-Lobo [2011].

Two alternatives were developed to select the next neighbor to explore. Both start with a

biased selection of the subset of products and periods according to probabilities $p(i) = \frac{\tau_i}{\sum_{j \in \mathcal{N}} \tau_j}$ for all products and $p(t) = \frac{\omega_t}{\sum_{l \in \mathcal{T}} \omega_l}$ for all periods. In the first option, which we call *P&MIP*^{rnd}, the products and periods subsets are selected just once defining the next neighbor to explore. In the second approach, *P&MIP*^{eval}, the selection of $\mathcal{N}'$ and $\mathcal{T}'$ is repeated K times. For each one of the K neighbors a single iteration of the inner loop in *SeqSearch* is performed to estimate the potential improvement that the neighbor can yield, which is inferred based on the obtained objective value of FS_r. Neighbors are then sorted in ascending order according to their potential. Let $\eta(k)$ be the rank of neighbor k. The probability $\mu(r)$ of choosing a candidate neighbor is given by:

$$\mu(r) = \frac{\eta(r)}{(K+1) \cdot K/2}.$$

For both cases, rather than having a random rule, we try to guide the search for the most promising neighbors. *P&MIP* ends according to the following stopping criteria: (1) the maximum running time allowed has been achieved, or (2) the maximum number of neighbors without improvement has been achieved in all neighborhood structures.

4. Computational results

In this section we present the computational experiments performed to validate our solution approach. In the following subsections instances are divided into three families according to their features. The first group of benchmark instances considers problem data having setup matrices obeying the triangular inequality. In the second instance group setups can violate the triangular inequality and, maximum and minimum lot sizes are also introduced. Finally, the last family of test instances is a collection of real-world problems.

All computational tests were conducted on Intel @ 2.40 GHz processing units limited to 4 GB of random access memory using the Linux operating system. The algorithm was implemented in C++ and compiled using a *gcc* compiler. IBM ILOG Cplex 12.1 was used both as the mixed integer and linear programming solver.

In all benchmark sets we use two variants of our solution approach to compare its performance against state-of-the-art algorithms or commercial solvers. Both apply *Relax-Price-Fix* to construct an initial solution to the problem and *Fix-Price-Optimize* as the improvement heuristic. The variants only differ in the neighbor selection step. The *P&MIP*^{rnd} variant selects the neighbors to explore based on their score, while *P&MIP*^{eval} selects neighbors to explore according to their potential (as described in Section 3.3).

The main goal of these computational experiments is to validate the approach under different problem settings, showing flexibility and robustness of the heuristic. Parameters were tunned during pre-testing and also reflect the empirical knowledge about the problem.

The following parameters values were used throughout the computational experiments. In *SeqSearch* we limit the sequence pool to ten times the number of products and prune the master solution once the percentage gap from the lower bound is below 0.01%. The *Relax–Price–Fix* construction heuristic takes two arguments: (1) σ - the number of time periods in each time partition and (2) β - the number of overlapping time periods between iterations. Through these parameters the number of iterations is automatically defined as $K = \lceil (T-\sigma)/(\sigma-\beta) \rceil + 1$. Preliminary tests of the construction heuristic led to $\sigma = 2$ and $\beta = 1$, as these values represent the best trade-off between efficacy and efficiency.

Both variants of our solution approach control the neighbors to be evaluated by the *Fix–Price–Optimize* heuristic, which requires a subset of periods and products to be re-optimized. The neighborhood structure definition, i.e. the number of periods and products to be solved, depends on the size of the instance. Table 1 presents the neighborhood structures defined. The following

Table 1: Neighborhoods used for *Fix–Price–Optimize*

	$T < 8$		$T \geq 8$	
Neighborhood	N'	T'	N'	T'
N_1	6	3	6	3
N_2	4	5	4	5
N_3	10	5	5	8

rule for the neighborhood structure definition was applied for all of the test instances and is derived from empirical studies during pre-testing. We always use 3 neighborhood structures, starting with one having a large subset of products and a small subset of adjacent time periods. In the subsequent neighborhood structure the periods subset is increased and the number of products to be re-optimize reduced. This represents the underlaying trade-off between the efficiency and effectiveness of the search. With the first neighborhood structures local minima is achieved as re-optimization is conducted for a small subset of periods, despite allowing for faster neighbor evaluations. Increasing the number of periods greatly increases the computational burden of the neighbor evaluation, although potentially allowing for a greater improvement of the incumbent solution. Hence, the increase in the number of periods is followed by a reduction in the number of products to smooth this impact. Finally, the last neighborhood structure attempts to escape from

close local minima by allowing changes in a large set of periods and products simultaneously. We allow up to 10 neighbors without improvement before moving to the next neighborhood structure.

4.1 Triangular setups

The first set of benchmark instances, available from James and Almada-Lobo [2011], assesses our solution approach under the presence of triangular setups. We use the data set related to single machine problems with capacity variation.

Problem instances are grouped into problems types defined by the quadruplet N, T, Cut, θ (representing: number of products, number of periods, average capacity utilization per period and cost of setup per time unit). A total of 240 instances were solved resulting from 10 different instances for each one of the 24 problem types created by combining the different values of the parameters: $N \in \{15,25\}$, $T \in \{5,10,15\}$, $Cut \in \{0.6,0.8\}$ and $\theta \in \{50,100\}$. For details concerning the problem instance generator the reader is referred to James and Almada-Lobo [2011].

In this benchmark, we compare the two variants of our solution approach with the Iterative Neighborhood Search heuristic starting with a Relax-and-Fix construction heuristic (INSRF) described in James and Almada-Lobo [2011], the best known method for the CLSD with triangular setups. All approaches have a maximum running time of one hour. We also present results for the construction heuristic *Relax–Price–Fix* (RPF). To evaluate the performance of the heuristics we calculate the gap from the lower bound reported in James and Almada-Lobo [2011]. A summary of the results is given in Table 2, which aggregates instances by each level chosen according to data parameters, e.g., row $N = 25$ aggregates all means obtained for instances with 25 products while the other parameters vary. We report the mean percentage gap from the lower bound, the average running times in seconds and the p-value resulting from the Student's t-test performed to validate the results (described later in this section).

These results validate the ability of our solution approach to successfully solve instances with triangular setups. As expected, the solution obtained by the construction heuristic is considerably improved by the neighborhood search. Generally, both variants have a lower mean gap than INSRF for harder problems, i.e. high number of products, high capacity utilization and high setup cost. The only exception occurs when the number of periods increases. For problems with fewer products, low capacity utilization and low setup cost the difference between the algorithms is less noteworthy, although for instances with a small number of periods INSRF is better than any of our solution approach variants. Clearly, the neighborhoods in INSRF are more effective for instances with a short planning horizon, while our solution approach is more effective to explore

Table 2: Summary of results for CLSD with triangular setups and capacity variation

		Mean gap (%)				p-value		Mean running time (s)			
		RPF	*P&MIP*^{rnd}	*P&MIP*^{eval}	INSRF	*P&MIP*^{rnd}	*P&MIP*^{eval}	RPF	*P&MIP*^{rnd}	*P&MIP*^{eval}	INSRF
N	15	1.76	1.34	**1.33**	**1.33**	0.480	0.499	85.2	246.0	315.1	2708.4
	25	2.10	0.94	**0.90**	0.99	0.113	*0.015*	601.1	1437.4	1854.4	1652.8
T	5	1.49	0.79	0.79	**0.67**	*≈ 0*	*≈ 0*	155.3	337.3	387.2	1140.6
	10	1.96	1.15	**1.14**	1.41	*≈ 0*	*≈ 0*	295.3	862.8	1054.9	3048.9
	15	2.34	1.47	1.42	**1.41**	*0.027*	0.364	578.9	1324.9	1812.0	2352.3
Cut	0.6	1.77	1.04	**1.01**	1.04	0.446	0.290	298.0	739.1	958.9	2004.6
	0.8	2.09	1.23	**1.22**	1.29	0.057	*0.021*	388.4	944.3	1210.5	2356.5
θ	50	0.97	0.35	**0.30**	0.31	*0.002*	0.361	190.2	436.5	796.4	1615.1
	100	2.89	**1.93**	1.93	2.02	*0.036*	*0.044*	496.2	1246.8	1373.0	2746.0
Overall mean		1.93	1.14	**1.12**	1.16	0.176	*0.041*	343.2	841.7	1084.7	2180.6

larger instances. Note that for $T = 15$, INSRF and *P&MIP*^{eval} produce almost the same gaps.

Neighbor selection also appears to play an important role in our solution approach. Guiding the partial neighborhood exploration process through the assessment of potential improvement of neighbors leads to lower mean gaps than by only combining neighbor scores and randomization.

To confirm these underlaying hypotheses of different performances of the tested heuristics, we carried out a paired Student's t-tests comparing the mean gaps of the two variants of *P&MIP* with INSRF for each one of the categories present in Table 2. The p-values reported refer to the alternative hypothesis that the method with lower mean gap has a better performance. In all cases *P&MIP* is considered, except for $T = 5$ and $T = 15$ where the alternative hypothesis is that the mean gap of INSRF is less than the mean gap of *P&MIP*. Considering a significance level of 0.05, the statistical tests confirm *P&MIP*^{eval} as the best approach for hard problems ($N = 25$, $\theta = 100$, $Cut = 0.8$). For easier instances we cannot draw conclusions. With respect to the mean gap, tests point *P&MIP*^{eval} as the overall best performing method and no statistical evidence of different performances between *P&MIP*^{rnd} and INSRF.

Finally, since the computational study of James and Almada-Lobo [2011] has been conducted on a similar computing architecture, using the same version of CPLEX, we assume that the running times are comparable. Both variants of our solution approach require considerable less computational time. The only exception are instances having a large number of products due to the increasing difficulty in solving the subproblem. Moreover, as expected, the neighbor evaluation step increases the solution time when compared to the selection based on a single sample.

4.2 Non-triangular setups

Next we present results concerning two benchmark sets in which setup matrices do not obey the triangular inequality. The first comes from the work of Kang et al. [1999] and enables us to validate the algorithm by benchmarking it against other solution procedures capable of tackling non-triangular setups. The second set is motivated by the small size of the instances in Kang et al. [1999]. To create larger non-triangular instances, we modify the problem set of James and Almada-Lobo [2011] by introducing shortcut products in the setup matrices, so that the triangular inequality does not hold anymore.

4.2.1 Small problems

Kang et al. [1999] created modified instances based on CHES problem number 5 (Baker and Muckstadt [1989]). A total of nine instances are available among which six are single machine problems *Da*,..., *Df*. All problems have six products and a planning horizon of nine time periods. Moreover, setup times are zero, products setups present a clustered structure and requirements on the maximum and minimum lot size are imposed. Different combinations of machine utilization and lot size requirements are used to generate the problem set (see Table 3). The two variants of our solution approach are compared to:

- **Kang**: the branch-and-price based heuristic of Kang et al. [1999];
- **Meyr SAPL**: the simulated annealing algorithm of Meyr [2002];
- **Meyr TAPL**: the threshold acceptance algorithm of Meyr [2002];
- **CPLEX**: Branch-and-Cut performed by parallel CPLEX 12.1 on the compact model for the CLSD with non-triangular sequence-dependent setups presented in Appendix B.

Besides the data features of the instances, Table 3 reports the upper bound provided by each method and the running times in seconds for the two variants of our solution approach and CPLEX. Both the upper bound and the running time presented in the case of our heuristic is the best run out of 20 different attempts (our heuristics embed a random component). We stress that Kang et al. [1999] and Meyr [2002] also report best values out of multiple runs.

Both variants of *P&MIP* were able to find the optimal solution for all problem instances (CPLEX 12.1 proves optimality in all instances). This validates the ability of the heuristic to deliver superior quality solutions even for easy instances. The heuristic strictly outperforms the

Table 3: Summary of results for the problems in Kang et al. [1999]

Instance		*Da*	*Db*	*Dc*	*Dd*	*De*	*Df*
Data features	Utilization (%)	95	99	70	95	99	95
	Minimum lot size	20	20	20	20	20	0
	Maximum lot size	200	200	200	100	100	1000
Upper bound	Kang	856.81	865.29	816.61	1263.01	1360.87	832.95
	Meyr TAPL	846.97	859.97	766.58	1182.11	1248.26	812.66
	Meyr SAPL	844.62	869.3	760.08	1174.01	1260.33	812.66
	P&MIP^{rnd}	842.64	852.32	758.49	1164.93	1202.33	812.66
	P&MIP^{eval}	842.64	852.32	758.49	1164.93	1202.33	812.66
	CPLEX	842.64	852.32	758.49	1164.93	1202.33	812.66
Deviation from Kang, Meyr	*P&MIP*^{rnd}	-0.2%	-0.9%	-0.2%	-0.8%	-3.7%	0.0%
	P&MIP^{eval}	-0.2%	-0.9%	-0.2%	-0.8%	-3.7%	0.0%
	CPLEX	-0.2%	-0.9%	-0.2%	-0.8%	-3.7%	0.0%

previous methods in terms of solution quality, except for problem D_f, for which the best solution previously reported is already optimal. The running times of our heuristic can not be compared to those of Kang, Meyr TAPL and Meyr SAPL since there are significant differences in hardware and software. However, we point out that CPLEX running times are shorter than the running times of our heuristic. In fact, the potential of our heuristic relies on solving bigger problems as exact methods are hard to beat for small instances like these ones. In the next section we report computational results on a set of larger instances to show this effect.

4.2.2 Modified triangular problems

Benchmark instances in the literature violating the triangular inequality are relatively scarce and small sized. The following benchmark set was designed in order to test the solution approach for harder instances of this type. Since hard instances for the triangular setup case are available in James and Almada-Lobo [2011], we chose to adapt them for the non-triangular setup case. To do so, we modify the setup time matrices to create a set $\mathscr{N}_{SC}$ of potentially shortcut products (products that lead to the violation of the triangular inequality). The number of shortcut products in the set is defined by $N_{SC} = \left\lceil \frac{N}{10} \right\rceil$. For each shortcut product $k \in \mathscr{N}_{SC}$ setup times st_{ik} and st_{ki} for each $i \notin \mathscr{N}_{SC}$ are generated from the uniform distribution between 2 and 4. The setup costs remain proportional to setup times using parameter θ.

Problems are classified as described in Section 4.1. We evaluate the performance of the two

variants of *P&MIP* against branch-and-cut performed by parallel CPLEX 12.1 on the compact formulation for the CLSD with non-triangular sequence-dependent setups presented in Appendix B. In order to evaluate the quality of the solutions generated by these procedures, we use the deviation from the best bound obtained by CPLEX during the tree search. Tables 4 and 5 present the results by problem type for low ($\theta = 50$) and high ($\theta = 100$) setup cost instances, respectively. Each instance is encoded under problem type as N-T-Cut-θ. Numbers in bold highlight the best average gaps. Problems types with five time periods were excluded, since they are too small.

Table 4: Summary of results for CLSD with non-triangular setups and low setup cost ($\theta = 50$)

	Mean deviation (%)			Mean running time (secs)		
Problem Type	CPLEX	*P&MIP*^{eval}	*P&MIP*^{rnd}	CPLEX	*P&MIP*^{eval}	*P&MIP*^{rnd}
15-10-0.6-50	**0.00**	0.22	0.45	58.8	876.9	955.0
15-10-0.8-50	**0.00**	0.53	0.67	434.6	986.9	1066.8
15-15-0.6-50	**0.00**	0.39	0.31	496.5	1334.2	1151.1
15-15-0.8-50	**0.04**	0.60	0.47	2138.2	1265.2	1225.5
25-10-0.6-50	**0.02**	1.60	1.40	1803.2	3601.5	3601.7
25-10-0.8-50	**0.03**	1.57	1.81	3057.5	3602.5	3601.4
25-15-0.6-50	**0.46**	2.25	2.12	3523.1	3602.2	3602.2
25-15-0.8-50	**0.25**	2.30	2.26	3601.9	3602.3	3602.7

Table 5: Summary of results for CLSD with non-triangular setups and high setup cost ($\theta = 100$)

	Mean deviation (%)			Mean running time (secs)		
Problem Type	CPLEX	*P&MIP*^{eval}	*P&MIP*^{rnd}	CPLEX	*P&MIP*^{eval}	*P&MIP*^{rnd}
15-10-0.6-100	**0.20**	1.06	1.36	2340.7	943.5	748.4
15-10-0.8-100	**0.76**	2.07	1.57	3600.3	1043.2	1086.1
15-15-0.6-100	**1.82**	2.20	2.07	3600.5	1214.9	1424.7
15-15-0.8-100	2.74	**2.39**	2.84	3600.4	1524.5	1186.2
25-10-0.6-100	**1.00**	2.15	2.34	3601.4	3601.6	3601.4
25-10-0.8-100	3.39	**2.35**	2.52	3601.4	3577.4	3601.3
25-15-0.6-100	6.82	2.67	**2.42**	3602.1	3602.1	3601.9
25-15-0.8-100[a]	9.79	**2.99**	3.24	3602.2	3602.2	3601.8

[a] CPLEX fails to achieve a feasible solution for 5 out of 10 instances in this problem type. Mean deviation were calculated based on the remaining instances.

These results indicate that instances with low setup cost ($\theta = 50$) are relatively easy for CPLEX, because it can often prove optimality or the final gaps are below 0.5%. Note that, in these settings, sequencing decisions are less important. For these instances our heuristic variants are not competitive. Nevertheless, the solution quality provided by the heuristics is never above 2.5% from the

solution found by CPLEX (most of the times the optimal solution), which indicates good quality solutions. Since CPLEX can often prove optimality, running times are shorter when compared to those of the heuristic variants.

For problems with high setup cost ($\theta = 100$), the heuristics reveal their efficiency. As the problems become harder (more products and/or time periods) the heuristics outperform CPLEX. This is the case for 15 products and 15 time periods, and 25 products and 15 time periods. This effect is particularly pronounced in the last two problem types shown in Table 5, for which CPLEX solution quickly deteriorates. In the last problem type, CPLEX even fails to deliver a feasible solution for 5 out of 10 instances, while our heuristics allways find a feasible solution. Comparing the two variants, the guided neighbor selection has an advantage over a single sample selection in terms of the solution quality.

For problem types with 15 products running times of the heuristics are shorter when compared to CPLEX, indicating an opportunity for additional improvements in the solution quality of the heuristics. For problems with 25 products, the searches end by the maximum running time allowed in any approach.

4.3 Real world instances

The last set of instances corresponds to a collection of seven different real-world problems from the beverage industry. Problems have a planning horizon of 8 weeks, common in tactical production planning in the process industry, while the number of products varies from 8 to 33. Problems are quite diverse, ranging from scenarios with high demand of few standard products to production lines dedicated to several products with low demand and highly customized. Across all instances, capacity is constant throughout the planning horizon, however orders are highly unbalanced between time periods.

Table 6 reports the results for the two variants of the heuristic and the Branch-and-Cut performed by parallel CPLEX 12.1 on the compact model for CLSD with non-triangular sequence-dependent setups presented in Appendix B.

Problems are sorted by the increasing number of products, which also increases the difficulty. For problems with less than 15 products CPLEX can obtain provably optimal solutions. Nevertheless, the percentage deviation of the solutions provided by our heuristics is quite small, attesting the ability of the heuristics in finding high quality solutions. For problems having a larger number of products, our heuristics outperforms the exact method. In these instances, CPLEX performance is not satisfactory as it even fails to deliver a feasible solution for two of the instances (S7 and L6,

Table 6: Summary of results for real-world instances

Instance			Objective			Deviation (%)			Time		
#	N	T	CPLEX a	*P&MIP*^{eval} b	*P&MIP*^{rnd} c	$(b-a)/a$	$(c-a)/a$	$(c-b)/b$	CPLEX	*P&MIP*^{eval}	*P&MIP*^{rnd}
S2	8	8	**100915.6***	101470.1	101470.1	0.55	0.55	0.00	132.77	182.13	108.68
L3	10	8	**94010.79***	94191.8	94090.05	0.19	0.08	-0.11	142.11	208.59	221.71
L5	11	8	**77514.12***	77833.78	77582.06	0.41	0.09	-0.32	192.69	153.76	162.43
S1	15	8	267409.2	183722.9	**183068.8**	-31.30	-31.54	-0.36	3600.31	998.35	1062.6
R2	19	8	**159439.4**	167705.6	168977.9	5.18	5.98	0.76	3600.73	1928.87	1100.3
S7	20	8	-	**189556.7**	199323.2	-	-	5.15	3600.74	3603.16	3507.41
L6	33	8	-	**464409.8**	489310.4	-	-	5.36	3603.11	3659.48	3611.29

* Optimal solution values

those with the largest number of products). Additionally, the solution found for instance S1 is 30% higher than the solutions of the heuristic variants.

Heuristic running times are competitive for easy problems compared to the exact method. In problems S1 and R2 we can observe that the neighborhood exploration process ends before the maximum running time limit, contrarily to CPLEX. For larger problems the search consumes the entire allowed running time in either case.

In terms of the two variants of the heuristic, there is not a clear difference in the first five problems. The benefits of the guided local search can only be seen in the last two instances, which have larger neighborhoods and, therefore, neighbor selection becomes more important. We observe an improvement of 5% in the solution quality for these cases.

5. Conclusions

In this paper we present a novel mixed integer programming formulation for the single machine capacitated lotsizing and scheduling problem. An important cause of computational intractability of large bucket models for lotsizing and sequencing often comes from the sequencing problem that has to be solved within each time period. The underlying idea of our model is to simplify the scheduling part by defining, for each time period, a limited pool of feasible production sequences. Afterwards, the model selects the most adequate one for each time period. The formulation handles non-triangular setups times and costs, as well as minimum and maximum lot sizes.

Based on the model we have developed a mathematical programming based solution approach that handles large size instances. It is composed of three main blocks: a pricing heuristic, a construction heuristic, and an improvement heuristic. The pricing heuristic, *SeqSearch*, is used to

manage the sequence pool of each time period, since a direct implementation considering all possible production sequences would require the use of an exponential number of variables. Both the construction and improvement heuristics are the outcome of combining the pricing heuristic with mathematical programming based heuristics. The key principle behind these procedures is the selection of smaller subproblems which are easier to tackle. The construction heuristic, *Relax-Price-Fix*, generates a feasible solution by introducing column generation within the relax-and-fix framework. A series of partially relaxed MIPs are solved until a feasible solution is obtained. An important innovation is the combination of a compact and extended formulation in a single model during the construction phase to relief the computational burden of the procedure. *Fix-Price-Optimize*, our improvement heuristic, escapes from local minima by re-optimizing a feasible solution over a small subset of the original problem. These partitions are explored in a systematic framework and *SeqSearch* is embedded during re-optimization to discover new production sequences.

In our computational tests we show the flexibility of using mathematical programming based heuristics by tackling different features of the problem. The benchmark sets solved ranged from the case of triangular setups to non-triangular setups with minimum and maximum lotsize restrictions. The approach outperformed the state-of-the-art algorithms for large problems. Using parallel CPLEX 12.1 on a compact formulation revealed to be the dominant method for small and low setup cost instances, but as the instances become bigger and harder both variants of *P&MIP* are a better approach. The results on the set of real-world problems solved have shown the practical application of the approach.

Our insights also indicate that to conduct a partial exploration of the neighborhood of an incumbent solution, devising a rule for neighbor selection has a positive effect regarding the quality of the solution.

One of the interesting topics for future research is the subproblem which arises in the pricing heuristic. In this paper, it is solved using exact approaches (branch-and-cut), however for instances having a large number of products to be scheduled, an approximate approach can save computational time. Even so, such an approach would slow the convergence of the column generation algorithm, hence the benefit is still to be studied and assessed. In addition, regarding the pricing heuristic, the diving scheme can be modified to improve the search for production sequences.

The techniques described in this paper aim to solve the problem under consideration, nevertheless most of the ideas can be applied to different lotsizing problems or to problems beyond this research subject.

Appendix A Imposing minimum and maximum lotsizes

Whenever non-triangular setups appear, minimum lot sizes must be imposed in the model to exclude solutions where 'empty' setups to shortcut or 'cleansing' are scheduled. Minimum lot sizes guarantee proper machine cleansing via the production of a minimum amount of a shortcut product. Furthermore, technological constraints may also impose a maximum lot size on each product run. Both of these features appear in the CHES instances (Baker and Muckstadt [1989]), which are a compilation of real world problems. To introduce lotsize requirements in our model, let min_i^l, max_i^l be the minimum and maximum lot size of each production run of product i, respectively. We also introduce decision variables X_{it}^a, X_{it}^b to be the production quantities of product i manufactured after and before the first setup occurs in period t, respectively. The following constraints are adapted from Menezes et al. [2011] and must be added to model FS.

$$\sum_{l \geq t} X_{itl} = X_{it}^a + X_{it}^b \quad \forall i \in \mathscr{N}, t \in \mathscr{T} \tag{A.1}$$

Minimum lot sizes:

$$X_{it}^b \leq \frac{cap_t}{p_i} \cdot Z_{it} \qquad \forall i \in \mathscr{N}, t \in \mathscr{T} \tag{A.2}$$

$$X_{i1}^b \geq min_i^l \cdot Z_{i1} \qquad \forall i \in \mathscr{N} \tag{A.3}$$

$$X_{it}^a \geq min_i^l \cdot (Y_{it} - Z_{it}) \qquad \forall i \in \mathscr{N}, t \in \mathscr{T} \setminus \{T\} \tag{A.4}$$

$$X_{it}^a + X_{i,t+1}^b \geq min_i^l \cdot Y_{it} \qquad \forall i \in \mathscr{N}, t \in \mathscr{T} \setminus \{T\} \tag{A.5}$$

$$X_{iT}^a \geq min_i^l \cdot Y_{iT} \qquad \forall i \in \mathscr{N} \tag{A.6}$$

Maximum lot sizes:

$$X_{it}^b \leq max_{it}^l \cdot Z_{it} \qquad \forall i \in \mathscr{N}, t \in \mathscr{T} \tag{A.7}$$

$$X_{it}^a + X_{i,t+1}^b \leq max_i^l \cdot Y_{it} \qquad \forall i \in \mathscr{N}, t \in \mathscr{T} \setminus \{T\} \tag{A.8}$$

$$X_{iT}^a \leq max_i^l \cdot Y_{iT} \qquad \forall i \in \mathscr{N} \tag{A.9}$$

Constraints (A.1) are commonly used to model minimum and maximum lotsize requirements and split the total period production into the amounts produced after and before the first setup is performed. Constraints (A.2)-(A.6) model minimum lot sizes. Constraints (A.2) impose that the production of a given product can only take place before the first setup if the product is carried over from the previous period. In (A.3) the minimum product lot is imposed for the initial setup configuration of the machine. Constraints (A.4) force production lots within the current period to respect the minimum lot size, while constraints (A.5) require that production within the current period plus the amount produced in the following period prior to the first setup must be at least

proportional to the minimum lot size and the number of setups performed to that product. Finally, constraints (A.6) are a special case of constraints (A.5) for the final period of the planning horizon. To model maximum lotsize constraints, (A.7)-(A.9) are necessary. Constraints (A.7) replace (A.2) with the same functionality and also restrict the amount produced before the first setup to the maximum of a production run. The maximum amount that can be produced considering the number of setups defined for the period is expressed by constraints (A.8), (A.9) for the first $T-1$ periods and for the final period, respectively. All these constraints rely on the assumption that at least a setup is performed in each time period. This implies that for some instances sub-optimal solutions are created. However, such cases are rare since we are modeling a big bucket problem, therefore is highly unlikely that production lots span more than one entire time period.

Appendix B A compact formulation for the CLSD with non–triangular setups

In order to formulate a new product related MIP model to theCLSD considerer T_{ijt} to be the number of changeovers from product i to product j in period t and G_{it} a binary variable indicating if product i is part of the production sequence in period t or not. Variables f_{ijkt} contain the flow of commodity k from node i to node j in period t and are used to prevent sub-tours. The model reads:

$$(FP) \quad \text{Min} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t\in\mathcal{T}} sc_{ij}\cdot T_{ijt} + \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}}\sum_{\substack{l\in\mathcal{T}\\ l>t}} (l-t)\cdot h_i \cdot X_{itl} \tag{B.1}$$

subject to:

$$\sum_{\substack{l\in\mathcal{T}\\ l\le t}} X_{itl} \ge d_{il} \qquad \forall i\in\mathcal{N}, l\in\mathcal{T} \tag{B.2}$$

$$\sum_{i\in\mathcal{N}}\sum_{\substack{l\in\mathcal{T}\\ l\ge t}} p_i\cdot X_{itl} + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} st_{ij}\cdot T_{ijt} \le cap_t \qquad \forall t\in\mathcal{T} \tag{B.3}$$

$$X_{itl} \le d_{il}\cdot G_{it} \qquad \forall i\in\mathcal{N}, t, l\in\mathcal{T}, l\ge t \tag{B.4}$$

$$Z_{it} + \sum_{j\in\mathcal{N}} T_{jit} = Z_{i,t+1} + \sum_{j\in\mathcal{N}} T_{ijt} \qquad \forall i\in\mathcal{N}, t\in\mathcal{T} \tag{B.5}$$

$$\sum_{i\in\mathcal{N}} Z_{it} = 1 \qquad \forall t\in\mathcal{T} \tag{B.6}$$

$$\sum_{j\in\mathcal{N}} T_{jit} + Z_{it} \ge G_{it} \qquad \forall i\in\mathcal{N}, t\in\mathcal{T} \tag{B.7}$$

$$\sum_{j\in\mathcal{N}} T_{jit} + Z_{it} \le setup_{it}^{max}\cdot G_{it} \qquad \forall i\in\mathcal{N}, t\in\mathcal{T} \tag{B.8}$$

$$f_{0jkt} \le Z_{jt} \qquad \forall j,k\in\mathcal{N}, t\in\mathcal{T} \tag{B.9}$$

$$f_{ijkt} \le T_{ijt} \qquad \forall i,j,k\in\mathcal{N}, t\in\mathcal{T} \tag{B.10}$$

$$\sum_{j\in\mathcal{N}} f_{0jkt} = G_{kt} \qquad \forall k\in\mathcal{N}, t\in\mathcal{T} \tag{B.11}$$

$$\sum_{j\in\mathcal{N}} f_{ijkt} = \sum_{j\in\mathcal{N}\setminus\{k\}} f_{jikt} + f_{0ikt} \qquad \forall k\in\mathcal{N}, i\in\mathcal{N}\setminus\{k\}, t\in\mathcal{T} \tag{B.12}$$

$$\sum_{j\in\mathcal{N}} f_{jkkt} + f_{0kkt} = G_{kt} \qquad \forall k\in\mathcal{N}, t\in\mathcal{T} \tag{B.13}$$

$$\left(X_{itl}, f_{ijkt}, f_{0jkt}\right) \ge 0, \quad (Z_{it}, G_{it}) \in \{0,1\}, \quad T_{ijt}\in\mathbb{N} \quad \forall i,j,k,\in\mathcal{N}\, t, l\in\mathcal{T}. \tag{B.14}$$

Objective function (B.1) minimizes the total expenditure in holding and setup costs. Constraints (B.2) guarantee demand fulfillment. The total production and setup time may not exceed the available machine capacity as ensured by constraints (B.3). Requirements (B.4) link products with the machine setup state. Constraints (B.5) balance the setup flow for each product. Equations (B.6) state that the machine is set up for exactly one product in the beginning of each time period. The relationship between the setup state and both the initial machine configuration and the changeovers are established by constraints (B.7) and (B.8). Constraints (B.8) also limit the number of changeovers to a given product in each time period by $setup_{it}^{max}$. Disconnected sub-tours are eliminated by constraints (B.9)-(B.13). Namely, constraints (B.9) and (B.10) ensure that the flows only traverse arcs in the period's sequence, while (B.11)-(B.13) impose that flow variables for each commodity k describe a path from the source (setup carry-over) to node k, if the node is present in the sequence of the respective time period.

Appendix C Subproblem formulation

The MIP model for the subproblem in time period t is as follows.

$$(sub_t) \quad \text{Min} \quad \sum_{i\in V}\sum_{j\in V} c_{ij}\cdot\chi_{ij} - \sum_{i\in\mathscr{N}} \rho_i \cdot y_i \tag{C.1}$$

$$\text{subject to:} \quad \sum_{j\in V}\chi_{ji} = \sum_{j\in V}\chi_{ij} \qquad \forall i\in\mathscr{N} \tag{C.2}$$

$$\sum_{j\in\mathscr{N}}\chi_{0j} = 1 \tag{C.3}$$

$$\sum_{j\in\mathscr{N}}\chi_{j,N+1} = 1 \tag{C.4}$$

$$\sum_{j\in\mathscr{N}}\chi_{ji} \geq y_i \qquad \forall i\in\mathscr{N} \tag{C.5}$$

$$\sum_{j\in\mathscr{N}}\chi_{ji} \leq setup_{it}^{max}\cdot y_i \qquad \forall i\in\mathscr{N} \tag{C.6}$$

$$y_i\in\{0,1\} \quad \chi_{ij}\in\mathbb{N} \qquad \forall i,j\in\mathscr{N} \tag{C.7}$$

Here c_{ij} and ρ_i are derived from pricing equation (11):

$$c_{ij} = sc_{ij} - st_{ij}\cdot\lambda_t - \pi_{jt}, \quad c_{0i} = -\theta_{it}^{f}, \quad c_{i,N+1} = -\theta_{i,t+1}^{l}, \quad \rho_i = \alpha_{it} \qquad \forall i,j\in\mathscr{N}.$$

Objective function (C.1) minimizes the cost of the traversed arc minus the prizes collected from the scheduled products (visited nodes). Constraints (C.2) balance in- and out-flow of each product. The source and sink nodes must be connected to a product, guaranteed by requirements (C.3) and (C.4), representing the first and last products in the sequence. Constraints (C.5) and (C.6) enforce the logical relationship between the arcs traversed and the products visited. Parameter $setup_{it}^{max}$ is an upper bound on the number of setups for product i in period t.

The model for the subproblem is, however, still incomplete, as a solution for (C.1)-(C.7) permits disconnected sub-tours. To eliminate such sub-tours we use multi-commodity-flow type constraints. Consider decision variables f_{ijk} as the flow of commodity k from the source to node k traversing arc (i,j), which is constrained to be 0 or 1. Furthermore, the additional decision variables $y_i^{'}$ equal to 1 in case node i is ever traversed. The difference between y_i and $y_i^{'}$ relies on the fact that the latter equals to one also if the product is scheduled immediately after the source (first in the sequence, not representing an actual setup into it). The following constraints are added to sub_t to prohibit disconnected sub-tours (adapted from Sarin et al. [2011]):

$$f_{ijk} \leq \chi_{ij} \qquad \forall i \in V,\, j,k \in \mathcal{N} \tag{C.8}$$

$$\sum_{j \in \mathcal{N}} f_{0jk} = y_k^{'} \qquad \forall k \in \mathcal{N} \tag{C.9}$$

$$\sum_{j \in \mathcal{N}} f_{ijk} = \sum_{j \in \mathcal{N} \setminus \{k\}} f_{jik} \qquad \forall k \in \mathcal{N},\, i \in \mathcal{N} \setminus \{k\} \tag{C.10}$$

$$\sum_{j \in V} f_{jkk} = y_k^{'} \qquad \forall k \in \mathcal{N} \tag{C.11}$$

$$y_i^{'} \leq y_i + \chi_{0i} \qquad \forall i \in \mathcal{N} \tag{C.12}$$

$$2 \cdot y_i^{'} \geq y_i + \chi_{0i} \qquad \forall i \in \mathcal{N}. \tag{C.13}$$

Constraints (C.8) ensure that flows only traverse the arcs in the solution. Constraints (C.9)-(C.11) require that flow variables for each commodity k describe a path from the source to node k, if node k is in the sequence defined by arc variables. In more detail, constraints (C.9) force a unit of flow to leave the source for each node in the sequence. Flow conservation constraints (C.10) are imposed for each node in the graph, and, finally, the flow of commodity k must reach node k by (C.11). The last two sets of constraints (C.12)-(C.13) represent the logical connections between node variables.

Acknowledgments

The first author is grateful to the Portuguese Foundation for Science and Technology for awarding him a grant (SFRH/BD/62010/2009).

References

B. Almada-Lobo, D. Klabjan, J. F. Oliveira, and M. A. Carravilla. Single machine multi-product capacitated lot sizing with sequence-dependent setups. *International Journal of Production Research*, 45(20):4873 – 4894, 2007.

F. Alvelos, A. de Sousa, and D. Santos. Searchcol: Metaheuristic search by column generation. In M. Blesa, C. Blum, G. Raidl, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 6373 of *Lecture Notes in Computer Science*, pages 190–205. Springer Berlin / Heidelberg, 2010.

T. Baker and J. Muckstadt. The CHES problems. Technical report, Chesapeake Decision Sciences, Inc., 1989.

E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

M. O. Ball. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16(1):21 – 38, 2011.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3): 316–329, 1996.

G. Bitran and H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1992.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151, 2011.

A. Clark, B. Almada-Lobo, and C. Almeder. Lot sizing and scheduling: industrial extensions and research opportunities. *International Journal of Production Research*, 49(9):2457–2461, 2011.

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

A. Federgruen, J. Meissner, and M. Tzur. Progressive interval heuristics for multi-item capacitated lot-sizing problems. *Operations Research*, 55(3):490–502, 2007.

D. Ferreira, R. Morabito, and S. Rangel. Solution approaches for the soft drink integrated production lot sizing and scheduling problem. *European Journal of Operational Research*, 196(2): 697–706, 2009.

B. Fleischmann and H. Meyr. The general lotsizing and scheduling problem. *OR Spektrum*, 19(1): 11–21, 1997.

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.

R. J. James and B. Almada-Lobo. Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers & Operations Research*, 38(12):1816 – 1825, 2011.

R. Jans and Z. Degraeve. Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(6):1619–1643, 2008.

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.

A. Kovács, K. N. Brown, and S. A. Tarim. An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups. *International Journal of Production Economics*, 118(1):282 – 291, 2009.

J. Krarup and O. Bilde. Plant location, set covering and economic lot size: An O(mn) algorithm for structured problems. In *Optimierung bei Graphentheoretischen und Ganzzahligen Probleme*, pages 155 – 180. BirkhauserVerlag, Berlin, 1977.

V. Maniezzo, T. Stützle, and S. Voß. *Matheuristics*, volume 10 of *Annals of Information Systems*. Springer US, 2010.

A. Menezes, A. Clark, and B. Almada-Lobo. Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. *Journal of Scheduling*, 14(2):209–219, 2011.

H. Meyr. Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, 120(2):311–326, 2000.

H. Meyr. Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, 139(2):277 – 292, 2002.

I. Muter, Ş. İ. Birbil, and G. Şahin. Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems. *INFORMS Journal on Computing*, 22(4):603–619, 2010.

Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

F. Sahling, L. Buschkühl, H. Tempelmeier, and S. Helber. Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, 36(9):2546–2553, 2009.

S. Sarin, H. Sherali, and L. Yao. New formulation for the high multiplicity asymmetric traveling salesman problem with application to the Chesapeake problem. *Optimization Letters*, 5(2):259–272, 2011.

L. A. Wolsey. Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. *Management Science*, 48(12):1587–1602, 2002.

X. Zhu and W. E. Wilhelm. Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38(11):987–1007, 2006.