Modelling Extensions and Hybrid Metaheuristics for the Capacitated Lotsizing and Scheduling Problem

António Luís Aroso Ribas de Menezes

September 2008

Acknowledgements

First of all, I would like to thank Professor Bernardo Almada-Lobo and Professor Alistair Clark for accepting me under their supervision. Their knowledge and expertise, along with their motivation and comprehension, have been most valuable to me and to the success of this thesis. It has been a great honor to work with them, sharing their ideas and their vision. Their passion and determination are unrivaled, and will inspire me throughout my personal and professional life.

I am most grateful to all the DEMEGI teachers, who have challenged my mind along the years, presenting me with new problems and giving me the opportunity to learn a lot on different and unexplored areas of interest.

I owe thanks to all my colleagues from MIEIG for the healthy competition and the support they have provided me in numerous occasions, and to all the other schoolmates that one way or another have helped me succeed and enjoy my time at FEUP, including those from TEUP with whom I have shared unforgettable moments. I am specially grateful to Desport, Inês, Susana, Pedro, André and Martinho from MIEIG, along with Diogo, Tiago, Matias and Salvaterra from TEUP for being much more than I could ever ask for.

I would also like to thank Joanne, Steve, Ash, Tom, Will and Marc for giving me a great time at Bristol.

Thank you Mãe, for making everything possible, and for the guidance that brought me where I am. Thank you Avô, for instilling me eagerness to know more, and for being my greatest inspiration.

Finally, thank you Joana. You have always been there, and I'm sure you'll always be.

Abstract

This work has been developed under the Dissertation Project discipline from the 5th year of the Integrated Master in Industrial Engineering and Management of the Faculty of Engineering of the University of Porto, in collaboration with the Bristol Institute of Technology from the University of the West of England. Its objective is to contribute to the development of production planning methods, focusing the Capacitated Lotsizing and Scheduling Problem (CLSP), a problem that tries to fulfill market demand while minimizing the total cost of production plans, heavily dependent on the production lots sequence and size.

Two approaches have been considered: a theoretical approach, to improve existing representation models, and a practical approach, to develop efficient solving methods.

In the theoretical field, a novel model for the CLSP was developed, which correctly handles two situations: on one hand, when setup times and costs do not respect the triangular inequality, a situation may occur where the optimal solution includes more than one batch of the same product in a single period - in other words, at least one sub-tour exists in the production sequence of that period. On the other hand, by allowing setup crossovers, flexibility is increased and better solutions can be found. In tight capacity conditions, or whenever setup times are significant, setup crossovers are needed to assure feasibility. This work has originated an article, submitted to a scientific journal and currently awaiting for revision.

In the practical field, a hybrid metaheuristic that relies on the improvement of existing solutions was developed and implemented. By taking information from incumbent solutions provided by the branch-and-cut algorythm, simplified sub problems that potentially lead to better solutions are constructed. Preliminary tests suggest that the method quickly improves the first incumbent candidates found. This is useful not only because good solutions

are quickly found, but also because upper bounds are quickly set in the branch-and-cut algorythm, thus pruning branches and decreasing the size of the explored tree. However, the proposed method fails to increase the overall efficiency of the branch-and-cut algorithm, raising several hypothesis. Still, important insights on hybrid metaheuristics were gained, benefiting further research.

Resumo

O presente trabalho foi realizado no âmbito da disciplina de Projecto de Dissertação do 5º ano do Mestrado Integrado em Engenharia Industrial e Gestão da Faculdade de Engenharia da Universidade do Porto, em parceria com o Bristol Institute of Technology da University of the West of England. Tem como objectivo contribuir para o desenvolvimento dos processos de planeamento de produção, focando o Capacitated Lotsizing and Scheduling Problem (CLSP), problema que visa satisfazer a procura e optimizar o custo total do plano de produção, fortemente influenciado pela sequência e dimensão dos lotes de produção.

Foram consideradas duas abordagens essenciais: uma abordagem teórica, com vista a melhorar os modelos de representação existentes, e uma abordagem prática, com vista a desenvolver métodos de resolução eficientes.

Do ponto de vista teórico, foi desenvolvido um novo modelo para o CLSP capaz de lidar simultaneamente com duas situações distintas: Por um lado, quando os custos e tempos de setup não respeitam a desigualdade triangular, pode aparecer uma situação onde a solução óptima inclui mais do que um lote de produção de um mesmo produto num único período - por outras palavras, existe pelo menos um sub tour na sequência de produção desse período. Por outro lado, ao permitir setup crossovers, aumenta-se a flexibilidade, permitindo a descoberta de melhores soluções. Em condições de capacidade apertada, ou quando os tempos de setup são muito significativos, os setup crossovers são necessários para garantir a existência de uma solução admissível. Deste trabalho resultou um artigo que presentemente se encontra submetido a uma revista científica da especialidade, aguardando revisão.

Do ponto de vista prático, foi desenvolvida e implementada uma metaheurística híbrida, assente no melhoramento de soluções existentes. Recorrendo à informação fornecida por soluções incumbentes provenientes do algoritmo branch-and-cut, são gerados sub proble-

mas simplificados que potencialmente originam melhores soluções. Resultados preliminares mostram que este método melhora rapidamente as primeiras soluções incumbentes encontradas. No entanto, não se verifica um aumento geral de eficiência do algoritmo branch-and-cut, ainda que a nova solução reduza o número de ramos explorados. Mesmo assim, várias observações pertinentes foram realizadas no âmbito das metaheuristicas híbridas, contribuindo para futuros trabalhos de pesquisa nesta área.

Contents

Chapter 1

Introduction

1.1 The Dissertation Project at the Faculty of Engineering at the University of Porto

This work has been developed under the Dissertation Project discipline from the 5th year of the Integrated Master in Industrial Engineering and Management of the Faculty of Engineering of the University of Porto, in collaboration with the University of the West of England.

With the Bologna Process, master degrees in Portugal mandatorily need to integrate a professional internship, subject to a final report, or a scientific dissertation. While the scientific dissertation has always been a possibility, traditionally, and since its origins, students from the Integrated Master in Industrial Engineering and Management programme of the Faculty of Engineering of the University of Porto have always concluded their studies with an individual, professional internship project in a company, having had the opportunity to integrate and apply the knowledge, skills and attitudes acquired throughout the programme to complex engineering problems.

Initial negotiations between Professor Bernardo Almada-Lobo from the Faculty of Engineering of the University of Porto and Professor Alistair Clark from the University of the West of England, aiming to establish future joint work projects, have created the opportunity for a graduate student to undergo a research project in both institutions, supervised by the two researchers. Thus, having successfully applied for the opportunity, the author has produced the first scientific dissertation project in the aforementioned context.

While initially aiming to obtain insights into the way different solution approaches must be combined to develop efficient tools for solving hard MIP problems, with special focus on the CLSP, the project has soon evolved into a whole different form: a mixed approach to the CLSP, consisting of a theoretical approach aiming to the development of exact formulations that correctly address complex real life situations, specifically the existance of non-triangular setup times and costs and the need for increased flexibility given by setup crossovers, and a practical approach, similar to the initial objective, aiming to the development of a hybrid metaheuristic that may help find better solutions based on the information contained on previously existing ones.

1.2 The Capacitated Lotsizing and Scheduling Problem

Manufacturing organizations are keen to improve their competitive position in the global marketplace by increasing operational performance. Production planning is crucial to this end and represents one of the most challenging tasks managers are facing today. The focus of the work is the short/medium-term scope, namely two of the most important and challenging production planning problems: lotsizing and scheduling.

When a machine is set up from a product to another, setup times and costs are incurred. Since these times and costs are often dependent of the production sequence, production scheduling has a direct impact in the overall production plan cost and in the production capacity available for lotsizing. In turn, inventory levels determined by lotsizing decisions and market demand also have a direct impact in the overall production plan cost. This way, production scheduling and lotsizing are intricately connected, and integrated approaches are needed to achieve the most efficient production plan possible.

Most models consider a finite planning horizon divided into discrete time periods. The single-stage, multi-item, lot-sizing and scheduling problem with capacity constraints is referred to as the capacitated lotsizing and scheduling 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. In practice, these problems are very complex and difficult to solve (NP-hard).

Usually, setups follow the so called *triangular inequality*, *i.e.*, for any three products, the cost and time required to directly set up the machine from one product to another is always less than the sum of those required when setting up via an intermediate product. However, several reasons may lead to the existence of non-triangular setup times or costs. For example, in some industries (chemical, pharmaceutical, food, dyeing, etc.), unwanted contamination occurs between certain products. To avoid it, additional cleansing operations must be performed during machine set ups, often requiring the use of expensive special products. Alternatively, products that absorb the contaminating substances (or even lower grade mixed products) can be produced in between, reducing setup times and costs. In this situation, a minimum lot size is often required for the intermediate product, so as to eliminate unacceptable contamination.

When the triangular inequality does not hold, it may be efficient to produce more than one batch of the same product in a given period. Though this concept is quite simple, it is also quite hard to model when considering sequence dependent setup times and costs, often leading to incomplete or inaccurate MIP formulations, and subsequent infeasible or sub optimal solutions. In this paper, a novel formulation that correctly handles this problem is proposed.

Setup times play a double role in production planning problems, because not only do they indirectly impact on the optimal solution's value (by constraining the inventory levels), but they also interfere with the solution's feasibility. In tight machine capacity situations, an effective distribution of setup times is required. Quite often, setup operations can be interrupted at the end of a period and resumed at the beginning of the next one with no additional prejudice, either due to the nature of the operations required or to the fact that there is no physical separation between periods. When this happens, a *setup cross over* or a $period - overlapping\ setup$ is said to exist.

A novel formulation for the CLSP which correctly handles non-triangular setup times and costs while enforcing the necessary feature of minimum lot sizes, and allows setup crossovers between adjacent periods was developed, and the resulting article, Menezes et al. [2008], was submitted to a scientific journal. To the best of our knowledge, we are the first to exactly model these features for the well known capacitated lotsizing and scheduling problem (CLSP), considered to be a big-bucket model as multiple products/setups may be produced/performed per planning period.

Additionally, due to the complexity of these problems, a hybrid approach combining heuristic and exact methods was developed and implemented for the original formulation. While preliminary tests show unsatisfactory results, important insights have been acquired, clarifying future research directions.

1.3 Literature Review

There is a vast amount of literature on lotsizing and scheduling models with setup times and costs. However, to the best of our knowledge, none is able to correctly handle both sequence dependent and non triangular setups for big-bucket models, let alone a combination of this and period overlapping setups.

Several contributions from the academic community have greatly improved the quality of CLSP models. Gopalakrishnan et al. [1995] are among the first to address the influence of significant setup times, proposing a model that handles setup carry overs for identical times and costs. Later, in Gopalakrishnan [2000], product dependent setup times and costs are considered. Sox and Gao [1999] propose a new model that only considers product dependent setup costs, but at the same time greatly reduces the number of binary variables, increasing the model's efficiency. Porkka and Kuula [2000] show that proper accounting for setup carryovers and setup times decrease the number of setups and frees a significant amount of production capacity. Suerie and Stadtler [2003] suggest a new model, considering both setup times and costs while keeping the number of binary variables low. Extensive computational tests prove the model's superiority with respect to previously existing models.

Haase and Kimms [2000] take a different approach, considering a CLSP model that handles sequence dependent setup times and costs, but by pre-defining efficient production sequences, sub-optimal solutions may be found. The authors also assume no inventory may exist at the beginning of the period in which a production lot of that same product is to be produced.

Clark and Clark [2000] model the CLSP with sequence-dependent setup times using a new mixed-integer programming formulation. They assume that up to a given number of setups occur in the time period between any two given products, independently of their demand patterns.

Almada-Lobo et al. [2007] propose two models that correctly handle sequence-dependent setup times and costs for large-bucket problems, but do not allow setup cross overs, and may result in sub optimal solutions when non triangular setup times and costs exist.

In Toso and Morabito [2005], periods are divided in subperiods, enabling the original big-bucket problem to be treated as a small-bucket one, where at most one setup may be performed per subperiod. This way, sequence dependent setup times (either respecting or not the triangular inequality) are correctly accounted for. However, the subdivision requires an *a priori* definition of the maximum allowed number of setup operations per period, thus heavily increasing the model size, or even demanding multiple tweaking experiments before obtaining the optimal solution. While minimum lot sizes are imposed, they only relate to the first subperiod after the machine has been set up, leading to potential sub-optimal solutions. Moreover, setup cross overs are not allowed, having to start and finish in the same period.

Suerie [2006] proposes a model that correctly handles setup cross overs, but only for small-bucket problems. A set of variables are introduced in relation to the standard model, keeping track of how much time each setup operation is performed in each period, the cumulative time a setup operation has been performed in any given period since the last time it started, and the availability of the machine at the beginning of every period (the machine is available for production if the entire setup operation has finished). Sung and Maravelias [2008] propose a similar model for large-bucket problems, with sequence independent setup times and costs.

New extensions driven by the need to model more realistic manufacturing problem settings demand the combination of existing algorithms, tighter models and stronger valid inequalities based on the polyhedral structure of these problems. This situation is not exclusive to production planning problems. In fact, several models from a wide range of research fields have been proven to be too complex to be optimally solved by state of the art optimization engines in a feasible amount of time, and have created the need for heuristic algorithms that, while not solving the problem to optimality, find reasonably good solutions in a reasonably low amount of time. However, reasonably good is frequently not enough, and powerful hybrid algorithms are often crucial to deliver superior solutions in reasonable computational times. Several authors have contributed to the development of such algorithms, either specifically designed to handle production planning problems, or designed to cope with a broader base of integer programming (IP) or mixed integer programming (MIP) problems.

Altough hybrid metaheuristics are a recent approach to complex problem solving, several authors have already contributed to the development of methods that handle general MIP problems, independently of their context.

Fernandes and Lourenço [2007] presents a simple algorithm for the job-shop scheduling problem that combines the GRASP local search heuristic with the branch-and-cut exact algorithm. The proposed method is compared with similar approaches, achieving overall better results.

In Danna et al. [2005], information contained in the continuous relaxation of a MIP problem is used to construct promising neighborhood structures that are then then formulated as MIP problems and solved recursively, through a method called Relaxation Induced Neighborhood Search (RINS). Additionally, a method that guides the MIP tree based on the similarity between each individual node and the best known feasible solution is proposed.

Fischetti and Lodi [2008] combine previously existing heuristics to find and improve initial solutions. In their approach, a feasibility pump (FP) technique is employed to discover a possibly feasible initial solution. This initial solution is later provided as the starting point to an improvement algorithm, local branching (LB). Since finding an initial feasible solution is sometimes hard and unecessary, the integration of both methods allows for the use of an initial unfeasible solution, whose feasibility is later recovered during the improvement method. Through computational tests, is is shown that such a combination is more efficient than traditional optimization engines and the use of the local branching technique alone.

Rothberg [2007] proposes a mixed approach, using an evolutionary algorithm for polishing mixed integer programming solutions. In this approach, MIP solutions are mutated and combined, and then used within a large neighborhood search framework. These techniques are then integrated within a MIP branch-and-cut framework, often finding significantly better feasible solutions to very difficult MIP problems.

Another proposal for a hybrid-metastrategy for combinatorial optimization problems, presented in Framinan and Pastor [2008], uses a Bound Driven Search algorithm that performs a local search to explore the most promising nodes, in a systematic "branch-and-bound" sense, being able to guarantee the optimal solution to a problem.

1.4 Thesis Synopsis

This work is divided into five chapters. The reminder of the thesis is as follows.

Chapter (2) presents a novel formulation for the capacitated lotsizing and scheduling problem with non triangular setups. We start by presenting the reader a previously built model that correctly handles sequence dependent setup times and costs. We then give a detailed technical explanation on the non triangular setup problem, and specify the constraint replacements and additions required to correctly handle it. Two complementing algorithms are proposed, so that the model can be dynamically implemented. The necessary feature of minimum lot sizes is then addressed, with the addition of several new constraints. We prove that the new model is superior to the previous one, and provide a small numerical example to compare results.

Chapter (3) presents a model that builds on the previous one to permit period overlapping setups. All necessary changes to the formulation are specified. A proof that the new model is even better than the one presented in Chapter (2) is supplied, and the aforementioned numerical example is again used to compare results.

In Chapter (4), the proposed hybrid metaheuristic is explained. We start by giving an overview of the hybrid meta-strategy, and proceed to detailing each individual step.

Finally, Chapter (5) summarizes our main findings, and hints future research directions.

Chapter 2

New Model for CLSP with Non Triangular Setup Costs and Times

2.1 Standard CLSP model with sequence dependent setup costs and times

Consider the following standard model for the CLSP with sequence-dependent setup costs and times, suggested by Almada-Lobo et al. [2007]. Here, t denotes time periods ranging from 1 to T, while i and j index the products, which are labeled from 1 to N. Furthermore, the set $\{1, 2, ..., M\}$ is denoted by [M]. A general single-stage model is considered, involving multiple items to be scheduled on a single machine with the following data:

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,
M_{it}	upper bound on the production quantity of product i in period t.

Binary variable T_{ijt} indicates whether or not a setup occurs on the machine configuration state from product i to j in period t. Continuous variable α_{it} keeps track of the machine state — if it is set up for product i (value 1) or not (value 0) — at the beginning of period t. Variable X_{it} represents the amount of product i to produce in period t, and I_{it} the stock of product i at the end of period t. Finally, auxiliary variable V_{it} ranks production lot of product i in period t, assuring that the machine is only set up for one product on any given time.

Lastly, v denotes optimal values of underlying optimization problems. This formulation, F_1, will be used as a starting point for the extensions presented later in this paper:

$$v(F_1) = \min \sum_i \sum_j \sum_t c_{ij} \cdot T_{ijt} + \sum_i \sum_t h_i \cdot I_{it} \tag{2.1}$$

$$I_{it} = I_{i(t-1)} + X_{it} - d_{it} \qquad i \in [N], t \in [T] \tag{2.2}$$

$$\sum_i p_i \cdot X_{it} + \sum_i \sum_j s_{ij} \cdot T_{ijt} \leq C_t \qquad t \in [T] \tag{2.3}$$

$$X_{it} \leq M_{it} \cdot \left(\sum_j T_{jit} + \alpha_{it} \right) \qquad i \in [N], t \in [T] \tag{2.4}$$

$$\sum_i \alpha_{it} = 1 \qquad t \in [T] \tag{2.5}$$

$$\alpha_{it} + \sum_j T_{jit} = \alpha_{i(t+1)} + \sum_j T_{ijt} \qquad i \in [N], t \in [T] \tag{2.6}$$

$$V_{it} + N \cdot T_{ijt} - (N-1) - N \cdot \alpha_{it} \leq V_{jt} \qquad i \in [N], j \in [N] \setminus \{i\}, t \in [T] \tag{2.7}$$

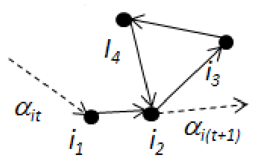
$$(X_{it}, I_{it}, \alpha_{it}, V_{it}) \geq 0, T_{ijt} \in \{0, 1\}. \tag{2.8}$$

The objective function (2.1) minimizes the sum of sequence-dependent setup costs and the holding cost. Constraints (2.2) represent the inventory balances and (2.3) ensure that production and setup operations do not exceed available capacity. Constraints (2.4) guarantee that a product is produced only if the machine has been set up for it. Constraints (2.5)-(2.7) determine the sequence of products on the machine in each period and keep track of the machine configuration state at the beginning of each period, by recording the product that a machine is ready to process at the end of the previous one (setup carryover information is thereby tracked).

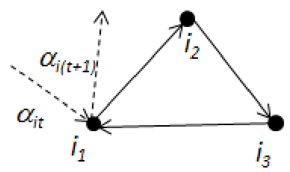
In practice, a situation may occur where more than one lot of the same product is produced in a single period. In other words, at least one *sub tour* – a production sequence that starts and ends in the same setup state – may exist in that period.

Two special sub tour cases are referred to throughout the article: *alpha* sub tours (sub tours that start and end in the same setup state as the first setup state of each period's production sequence) and *disconnected* sub tours (sub tours that are not part of the period's main sequence). Disconnected sub tours are further classified according to their complexity: *simple* disconnected sub tours (sub tours that form a perfect loop) and *complex* disconnected sub tours (sub tours that in turn are formed by multiple sub tours).

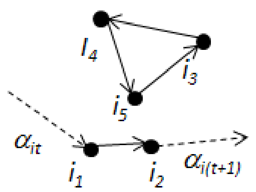
Consider a digraph G where nodes represent production lots of product i, solid arcs (i, j) represent setups from product i to product j, and dashed arcs represent the setup states inherited from or passed to neighboring periods, thus producing a visual representation of the production sequence of a given period. Figure 2.1 shows some sub tour examples, including the aforementioned special cases.



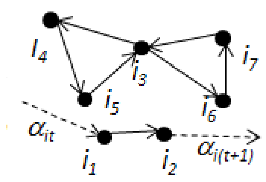
(a) Main sequence with regular sub tour



(b) Main sequence with alpha sub tour



(c) Main sequence and simple disconnected sub tour



(d) Main sequence and complex disconnected sub tour

Figure 2.1: Sub tour examples

It is evident that disconnected sub-tours cannot be part of a feasible solution, as it is impossible to define a finite chronological sequence that represents them correctly. The following reasoning shows that constraints (2.7) eliminate all but alpha sub tours.

Let $C \subseteq [N]^2$ represent a non-empty subset of product pairs (i, j), with cardinality $|C|$. If $T_{ijt} = 1$, $\forall (i, j) \in C$ in a given period t, then we say C represents a sub tour (or a group of sub tours). By summing constraints (2.7) up for all $(i, j) \in C$, we get $\sum_C V_{it} + N \cdot \sum_C T_{ijt} - \sum_C (N - 1) - N \cdot \sum_C \alpha_{it} \leq \sum_C V_{jt}$. Considering a *non-alpha* sub tour, we get $\sum_C \alpha_{it} = 0$. Evidently, $\sum_C V_{it} = \sum_C V_{jt}$, $N \cdot \sum_C T_{ijt} = N \cdot |C|$ and $\sum_C (N-1) = N \cdot |C| - |C|$. Thus, we get $N \cdot |C| - N \cdot |C| + |C| - 0 \leq 0$, which is clearly impossible for $|C| > 0$. This guarantees no sub tour occurs, with the exception of *alpha* sub tours.

2.2 Allowing sub tours in the main sequence

The following changes must be made to F_1 to correctly account for non-triangular setup costs and times:

Firstly, since non-triangular setup costs and times may result in a given setup being performed more than once, variables T_{ijt} must be allowed to take any non-negative integer values.

Secondly, constraints (2.7) must be replaced, as they only allow alpha sub tours. Let M represent a very large number, $S \subseteq [N]$ be a non-empty, non-unitary subset of the entire products set, and Y_{it} be a binary variable taking the value of 1 when the machine is is configuration state for to product i at least once in period t, and 0 otherwise:

$$Y_{it} = \begin{cases} 1 & \text{if } \sum_j T_{jit} + \alpha_{it} \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The following constraints are valid for any feasible solution (with or without sub tours), and cut off disconnected sub-tours:

$$\sum_{j \notin S} \sum_{i \in S} T_{jit} + \sum_{i \in S} \alpha_{it} + M \cdot \sum_{i \in S} (1 - Y_{it}) \geq 1 \qquad t \in [T], S \subseteq [N], |S| \geq 2 \tag{2.9}$$

These constraints are non-active whenever the machine is not configured to produce at least one product i in S in period t. If this is not the case ($Y_{it} = 1$ for every $i \in S$), then

(2.9) reduces to $\sum_{j \notin S} \sum_{i \in S} T_{jit} + \sum_{i \in S} \alpha_{it} \geq 1$. Clearly, this expression assures that the number of *inward links* (setups from another production lot, or the period's beginning) to a given set of production lots is always greater than or equal to one, as exemplified in Figure 2.2). If S represents a productive sub tour, it forces S's cycle to be connected to the production sequence of the previous period (through α's) or to the main sequence of that period (through T's), therefore it cuts disconnected sub tours off. Note that all regular sub tours (even non-alpha ones, such as S_1 on Figure 2.2) are allowed by (2.9).

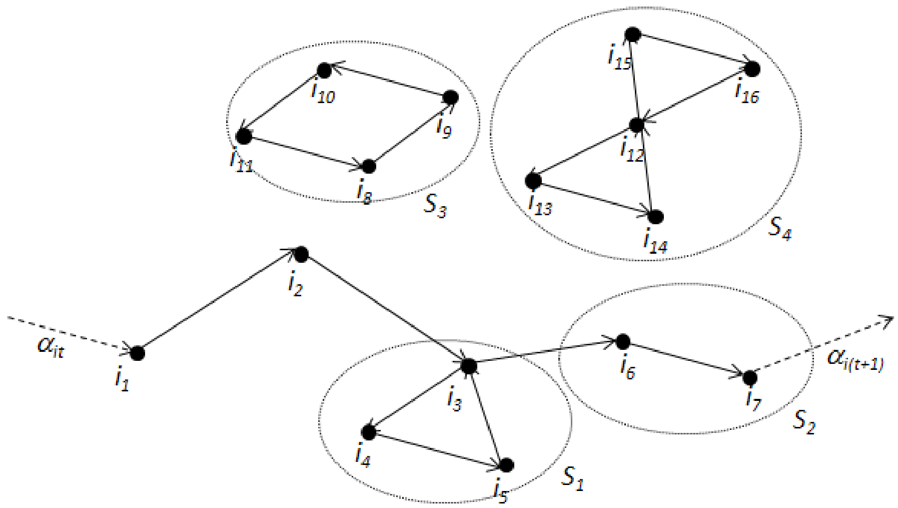


Figure 2.2: Examples of S: regular sub tour (S_1), part of the main sequence (no sub tour exists, S_2), simple disconnected sub tour (S_3) and complex disconnected sub tour (S_4)

The set (2.9) results in $T \cdot 2^N$ constraints, making full implementation impracticable. As such, individual constraints (2.9) will be dynamically added, as opposed to including them directly in the initial model. This can be done in a number of ways. In this article, we propose adding such constraints during the branch-and-cut process, whenever a disconnected sub tour is identified, thus removing it and preventing it thereafter. Two algorithms are used to achieve this end.

Algorithm 1, *FindST*, identifies a set of disconnected production lots S (which either form a disconnected sub tour or a group of disconnected sub tours).

The sub tour identification algorithm is called at every feasible node during the branch-and-cut process. If any disconnected sub tour is found, a global cut will be added, eliminating

```
for t ← 1 to T do
    for i ← 1 to N do
        if α_it = 1 then
            Connected_it ← 1;
            Validate(i,t);
            break out of i loop;
        end
    end
    for i ← 1 to N do
        if Connected_it = 0 and Y_it = 1 then a non-validated production lot exists
            Disconnected_it ← 1;
        end
    end
end
AddConstraints(Disconnected);
```

Algorithm 1: The sub tour identification (*FindST*) algorithm

and preventing it from happening again. In order to find production lots not linked to the main sequence, *FindST* algorithm calls recursive *Validate* algorithm several times (see Algorithm 2)

The recursive product validation algorithm works like this: Given a product i that is known to be part of the main sequence, setups from i to every other product j are checked. For each j, if at least one setup occurs from i to j ($T_{ijt} \geq 1$), and j is so far not known to be part of the main sequence, then j is *validated* as being part of the main sequence, and the product validation algorithm is called again, with j as argument. The product validation algorithm is initialized in every period with the first product in sequence as argument (given by $arg_i(\alpha_{it} = 1)$)

At the end of the process, production lots that are not validated as being part of the period's main sequence, $S_t = \{args_i(Disconnected_{it} = 1)\}$, are known to be disconnected, and the corresponding constraint is added.

Note that this approach intends to identify and remove any disconnected sub tour en-

```
Input: Connected product i to explore
Input: Period t
for j ← 1 to N do
    if Connected_jt =0 then
        if T_ijt ≥ 1 then
            Connected_jt ← 1;
            Validate(j,t);
        end
    end
end
```

Algorithm 2: The recursive product validation (*Validate*) algorithm

countered. However, it is also possible to use additional *a priori* polynomial sized constraints that prevent *simple* disconnected sub tours (thus reducing the number of dynamically added constraints) as follows:

Let a new binary variable Q_{ijt} be 1 if at least one setup operation T_{ijt} is performed, and 0 otherwise:

$$Q_{ijt} = \begin{cases} 1 & \text{if } T_{ijt} \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The following modification of constraint (2.7) allows connected sub tours, while removing simple disconnected ones:

$$V_{it} + M \cdot (Q_{ijt} - 1) + M \cdot \left(Q_{ijt} - \alpha_{it} - \sum_{l} T_{lit} \right) \leq V_{jt} - 1 \quad i \in [N], j \in [N], t \in [T] \tag{2.10}$$

Constraints (2.10) works as follows:

Consider a general cycle $C \in [N]^2$ of size $|C|$, with $T_{ijt} \geq 1, \forall (i,j) \in C$, which may be composed of single or multiple sub tours, and let S be the node set of C. Summing all the constraints (2.10) up for every arc (setup) belonging to cycle C, we obtain the following requirement:

$$|C| \leq M \cdot \sum_{(i,j)\in C} \left(\alpha_{it} + \sum_{l} T_{lit} - 1 \right) \tag{2.11}$$

If cycle C corresponds to a simple disconnected sub tour, it is evident that $\sum_l T_{lit} = 1$ and $\alpha_{it} = 0$, $\forall i \in S$, which violates constraints (2.11). In any other case, $\alpha_{it} + \sum_l T_{lit} \geq 2$ for the sub tour *joint* (the product that starts and ends a regular (product i_2 in Figure 2.1a) or alpha (product i_1 in Figure 2.1b) sub tour, or bridges multiple sub tours into a complex disconnected sub tour (product i_3 in Figure 2.1d)), thus fulfilling the imposed requirements.

2.3 Enforcing minimum lot sizes

In cases where non-triangular inequalities exist due to the possibility to produce intermediate lower-grade or cleansing products, minimum lot sizes must be imposed, so as to guarantee an effective machine cleansing. Data representing the minimum lot sizes of each product should be added to the model:

m_i minimum size of each production lot of product i.

Additionally, a new binary variable R_t that equals to one if at least one setup is performed during period t is required:

$$R_t = \begin{cases} 1 & \text{if } \sum_i \sum_j T_{ijt} \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The following constraints are added:

$$X_{it} = X_{it}^{-1} + X_{it}^{0} \qquad i \in [N], t \in [T] \tag{2.12}$$

$$X_{it}^{-1} \leq M \cdot \alpha_{it} \qquad i \in [N], t \in [T] \tag{2.13}$$

$$X_{it}^{0} \geq m_i \cdot \left(\sum_j T_{jit} - \alpha_{i(t+1)} \right) \qquad i \in [N], t \in [T] \tag{2.14}$$

$$X_{it}^{0} + \sum_{k=t+1}^{s} X_{ik}^{-1} \geq m_i \cdot \sum_j T_{jit} - M \cdot \left(\sum_{k=t+1}^{s-1} R_k + 1 - R_s \right) \qquad i \in [N], t \in [T], s \in [T] \backslash [t] \tag{2.15}$$

Expressions (2.12) split production X_{it} into the amount X_{it}^0 of product i produced in period t after setups are performed in that period, and the amount X_{it}^{-1} produced at the beginning of the period, after a setup carry-over. If the setup state of product i is not carried over into period t ($\alpha_{it} = 0$), then clearly $X_{it}^{-1} = 0$, as imposed by (2.13). Constraints (2.14) assure production lots that start and end within period t fulfil the minimum lot size requirement. Constraints (2.15) enforce a minimum production size proportional to the number of setups to product i in period t, allowing that same production size to be split into subsequent periods. Note that this will only be enforced if there is at least one setup occurring in period s (*i.e.*, the cross over production lot ends), and there are no setups between $t+1$ and $s-1$ (*i.e.*, a unique cross over production lot is being considered). This can be simplified if production lots never span for more than one entire period (*i.e.*, $s = t+1$ instead of $s \in [T] \backslash [t]$).

Finally, variable domains must be specified:

$$(X_{it}, X_{it}^0, X_{it}^{-1}, I_{it}, \alpha_{it}, V_{it}) \geq 0, T_{ijt} \in \mathbb{N}_0, (Y_{it}, R_t) \in \{0, 1\} \tag{2.16}$$

The new formulation, F_2, consists of objective function (2.1) subject to constraints (2.2)-(2.6), (2.9) and (2.12)-(2.16). We prove in the following lemma that F_1 is a special case of F_2, and, as such, the optimal solution of F_2 is at least as good as F_1's:

Lemma 1. $v(F_1) \geq v(F_2)$

Proof. F_2 can be seen as a generalization of F_1, since the latter can be derived from the former by adding additional constraints. If minimum lot sizes are not enforced, by setting $m_i = 0$, $\forall i \in [N]$, then constraints (2.12)-(2.15) become redundant and can be dropped. In addition, let the set

$$\sum_{(i,j)\in C} \alpha_{it} \geq \frac{|C| \cdot (1-N)}{N} + \sum_{(i,j)\in C} T_{ijt} \tag{2.17}$$

be added to F_2, obtained by all the constraints (2.7) listed for every arc (i, j) of a general cycle C. As $\sum_{(i,j)\in C} T_{ijt} \geq |C|$, $\frac{|C|}{N} > 0$ and α_{it} can only take on integer values, then (2.17) reduces to $\sum_{(i,j)\in C} \alpha_{it} \geq 1$. If S contains the node set of cycle C, then (2.9) is equivalent to $\sum_{i\in S} \alpha_{it} \geq 1 - \sum_{j\notin S} \sum_{i\in S} T_{jit}$. Clearly, (2.17) makes this constraint redundant as $\sum_{i\in S} \alpha_{it} = 1$ if $\sum_{(i,j)\in C} \alpha_{it} \geq 1$. In the presence of sub tours, (2.17) dominates (2.9), making the feasible solutions of F_2 with (2.17) coincident with those of F_1. Therefore, the set of feasible solutions of F_1 is a subset of the set of feasible solutions of F_2. Consequently, for the same data set, $v(F_1) \geq v(F_2)$, completing the proof. □

The following example demonstrates the previous statement by showing the optimal solutions of the same instance to F_1 and F_2

Example 1. A production plan of five different products, $i = \{1, 2, 3, 4, 5\}$, over the next three periods must be devised. A certain component of product 5 contaminates product 1, and an expensive disinfectant product is required to clean the machine, thus increasing c_{51}. Product 3 has a component that absorbs the contaminating component from product 5, hence the triangular inequality will not hold for the setup costs of sequence $5-3-1$. Table 2.1 shows the relevant data for this problem. Additionally, consider $c_{ij} = 10 s_{ij}$, $\forall (i, j) \setminus (5, 1)$, $c_{51} = 250$, and $C_t = 100$, $\forall t$.

Tables 2.2 and 2.3 show the most relevant non-zero solution values given by F_1 and F_2, respectively. Those same solutions are graphically represented by figures 2.3 and 2.4. Here, white blocks represent production that is to be consumed in that period, light grey blocks represent production that is to be stocked, middle grey represents idle time and dark grey represents setups.

Table 2.1: Data for the five product, three period problem

	d_{it}			s_{ij}					
	$t=1$	$t=2$	$t=3$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	h_i
$i=1$	90	0	110	-	20	100	100	100	10
$i=2$	0	10	0	100	-	5	100	100	10
$i=3$	0	10	0	10	100	-	10	100	10
$i=4$	0	10	0	100	100	100	-	10	10
$i=5$	0	10	0	10	100	5	100	-	10

Table 2.2: F_1's optimal solution

$t=1$	$t=2$	$t=3$
$\alpha_{11}=1$	$\alpha_{12}=1$	$\alpha_{13}=1$
$X_{11}=95$	$X_{12}=5$, $T_{122}=1$	$X_{13}=100$
	$X_{22}=10$, $T_{232}=1$	
	$X_{32}=10$, $T_{342}=1$	
	$X_{42}=10$, $T_{452}=1$	
	$X_{52}=10$, $T_{512}=1$	
$I_{11}=5$	$I_{12}=10$	

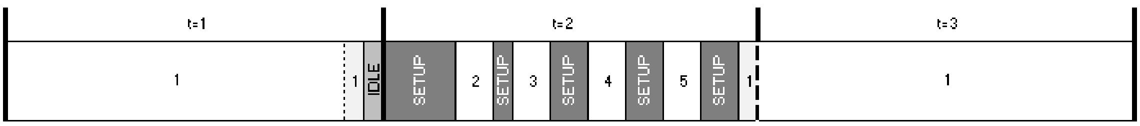


Figure 2.3: Graphical representation of F_1's optimal solution

Note that in F_1's optimal solution, 5 units of product 1 must be added to stock in periods

1 and 2 to fulfill demand in period 3, with a holding cost of 150 monetary units. Setup costs account for 700 monetary units, resulting in an objective function value of 850 monetary units.

Table 2.3: F_2's optimal solution

$t = 1$	$t = 2$	$t = 3$
$\alpha_{11} = 1$	$\alpha_{12} = 1$	$\alpha_{13} = 1$
$X_{11} = 100$	$T_{122} = 1$	$X_{13} = 100$
	$X_{22} = 10$, $T_{232} = 1$	
	$X_{32} = 10$, $T_{342} = 1$, $T_{312} = 1$	
	$X_{42} = 10$, $T_{452} = 1$	
	$X_{52} = 10$, $T_{532} = 1$	
$I_{11} = 10$	$I_{12} = 10$	

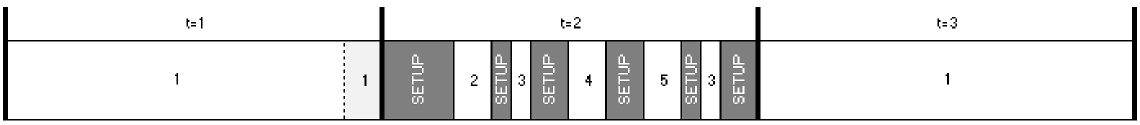


Figure 2.4: Graphical representation of F_2's optimal solution

In F_2's optimal solution, an extra setup exists. However, due to the non triangular inequality of sequence $5 - 3 - 1$, total setup costs are reduced by 100 monetary units, to a total cost of 600. The increase in setup times force production of product 1 in period 2 to be anticipated to period 1, increasing holding costs by 50 monetary units, to a total of 200. This results in an objective function value of 800 monetary units, which is 50 less than F_1's. Note that our solution does not include the size of each individual production lot, but instead the total amount of each product to be produced in each period. The example depicted in Figure 2.4 represents one of the many possible ways to split $X_{32} = 10$ units of product 3 between the two corresponding production lots in period 2. These variations have no impact in the objective function value.

Chapter 3

New Model for CLSP with Sequence-dependent and Period Overlapping Setup Costs and Times

Both F_1 and F_2 only take into account solutions that entail setups performed entirely within a time period. We now consider lotsizing and scheduling problems where setups are allowed to overlap period's boundaries. Such feature is of upmost importance to tackle tight capacity scenarios.

Our model uses two new types of variables (one of them binary) in addition to the variables from model F_2. Continuous variables S_t contain the amount of time still needed to finish the last setup operation at the end of period t (cross over time). Binary variables B_{ijt} indicate whether or not the cross over setup from period t to period $t+1$ is from product i to j.

Due to setup cross overs, setup times S_t that are delayed to the following periods (as well as setup times S_{t-1} that are inherited from previous ones) must be taken into account. Thus, capacity constraints (2.3) must be extended in the following way:

$$\sum_i p_i \cdot X_{it} + \sum_i \sum_j s_{ij} \cdot T_{ijt} - S_t + S_{t-1} \leq C_t \quad t \in [T] \tag{3.1}$$

To ensure setup cross overs S_t only occur if a given setup T_{ijt} is performed and that they do not exceed the corresponding setup time s_{ij} we add:

$$S_t \leq \sum_i \sum_j s_{ij} \cdot B_{ijt} \quad t \in [T] \tag{3.2}$$

$$B_{ijt} \leq Q_{ijt} \quad i \in [N], j \in [N], t \in [T] \tag{3.3}$$

To ensure only the last setup performed may cross over, we add:

$$\sum_j B_{jit} \leq \alpha_{i(t+1)} \quad i \in [N], t \in [T] \tag{3.4}$$

Note that since $\sum_i \alpha_{it} = 1$, we have $\sum_i \sum_j B_{ijt} \leq 1$, which prevents multiple setups from crossing over.

Constraints (2.4) must be extended so that production of product i in period t may only occur if at least one full setup operation for that product ends in that period.

$$X_{it} \leq M_{it} \cdot \left(\sum_j (T_{jit} - B_{jit}) + \alpha_{it} \right) \quad i \in [N], t \in [T] \tag{3.5}$$

Note that this constraint may become very loose, but capacity constraints and inventory costs will always prevent X_{it} from getting too big.

Minimum lot sizes must be enforced when a setup crosses over:

$$\sum_{k=t+1}^{s} X_{ik}^{-1} \geq m_i \cdot \sum_j B_{jit} - M \cdot \left(\sum_{k=t+1}^{s-1} R_k + 1 - R_s \right) \quad i \in [N], t \in [T], s \in [T], s > t \tag{3.6}$$

Finally, we assure S_t is non-negative, and Q_{ijt} and B_{ijt} are binary:

$$S_t \geq 0, (Q_{ijt}, B_{ijt}) \in \{0, 1\}. \tag{3.7}$$

The new formulation, F_3, consists of objective function (2.1) subject to constraints (2.2), (2.5), (2.6), (2.9), and (2.12)–(3.7).

Let S_2 and S_3 be the sets of feasible solutions to F_2 and F_3, respectively. We prove in the following lemma that F_2 is a special case of F_3.

Lemma 2. $S_2 \subseteq S_3$.

Proof. Let us assume another model and its feasible solution set, F_3* and S_3*, respectively, similar to F_3 with the following additional requirement:

$$\sum_i \sum_j \sum_t B_{ijt} = 0 \tag{3.8}$$

This constraint ensures no setup cross over occurs. Constraints (3.2) make variables S_t all equal to 0. It becomes obvious that F_3* is equivalent to F_2, and therefore $S_2 = S_3*$. Since F_3* is a restricted version of F_3, we can conclude that F_3 admits all of F_3*'s feasible solutions, *i.e.*, $S_3* \subseteq S_3$, which is equivalent to $S_2 \subseteq S_3$. □

The following example demonstrates that F_3 can achieve a better optimal solution than F_2:

Example 2. Consider the same data set of Example 1. Figure 3.1 shows the optimal solutions to F_3.

Table 3.1: F_3's optimal solution

$t = 1$	$t = 2$	$t = 3$
$\alpha_{11} = 1$	$\alpha_{22} = 1$	$\alpha_{13} = 1$
$X_{11} = 90$, $T_{121} = 1$	$X_{12} = 10$	$X_{13} = 100$
$B_{121} = 1$, $S_1 = 10$	$X_{22} = 10$, $T_{232} = 1$	
	$X_{32} = 10$, $T_{342} = 1$, $T_{312} = 1$	
	$X_{42} = 10$, $T_{452} = 1$	
	$X_{52} = 10$, $T_{532} = 1$	
	$I_{12} = 10$	

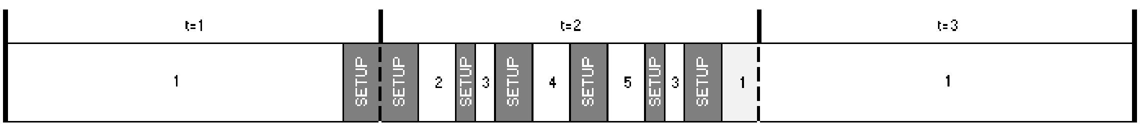


Figure 3.1: Graphical representation of F_3's optimal solution

By allowing the first setup to cross over, it is possible that the extra 10 (inventory) units of product 1 that are being produced in period 1 in F_2's optimal solution are pushed into period 2, reducing holding costs by 100 monetary units, while keeping the same setup costs. This results in an objective function value of 700 monetary units, which is 100 less than F_2's.

Chapter 4

Hybrid Metaheuristic for the CLSP

The hybrid metaheuristic developed throughout this project relies on two assumptions.

Firstly, we assume that sub-optimal feasible solutions can be improved by optimally solving small portions of it. In other words, it is sometimes possible to obtain better solutions by fixing part of the decision variables and optimally solving the remaining problem. This reduced sub-MIP, although in many cases not able to provide the optimal solution to the whole problem, can be solved considerably faster, quickly improving or accepting the best solution found so far in a reduced neighborhood.

Secondly, we assume that as solutions get better (i.e., their objective function value becomes closer to the optimum), they are likely to get similar to the optimal solution. This is especially accurate when better solutions are found by adding restrictions to the solutions space, as happens throughout the branch-and-cut algorithm: restrictions decrease the solution space, hence reducing diversity. As an example, we can argue that, in the context of the CLSP, the most expensive and time consuming setup operations will most likely not be part of the optimum solution, and, as better options are readily available, they will quickly be discarded from the search tree. It is therefore plausible to say that once a certain parcel of the incumbent solution matches the same parcel of the optimal solution, new incumbents will keep that same match. Hence, one can argue that when a pattern is found in successive solutions (i.e., the solution value of a certain decision variable is the same), that pattern is likely to be part of the optimal solution.

Combining both assumptions, we argue that when a pattern is found over successive incumbent solutions, focus should be put on the remaining segments of the solution. Exploiting this combined assumption, our method identifies pattern information from previous incumbent solutions, and uses it to construct simplified sub problems that potentially lead to better solutions.

We begin by taking running the branch-and-cut algorithm. Whenever an incumbent solution is found, a pattern identification method is applied – quite simply, a variable array containing the exponential smoothing of the binary variables that are part of previous solutions, *smooth*, is updated. This way, each element of the *smooth* array will indicate wether or not the associated variable has a tendency to a certain value (0 or 1). In this step, the use of exponential smoothing is preferred to the average, so as to give more relevance to recent (better) results, while at the same time reacting more quickly to pattern changes.

After updating the entire array, a constructive mask for the sub-MIP is defined. For every binary variable, it is randomly decided whether or not fixing bounds are set. The decision is based on a probability factor, f. If a random number between 0 and 1 is smaller than f, the variable is fixed, otherwise it becomes a decision variable for the sub-MIP. Allowing any variable to be unbounded in the sub-MIP, even if a strong pattern is detected, originates a certain degree of diversity in new solutions. While relatively small, this diversity may some times lead to the discovery of "hidden" solutions, that do not respect our second assumption.

Having both the mask and the *smooth* array, the sub-MIP is build – each bounded variable is assigned the value of 1 if a number between 0 and 1 is smaller than the corresponding value in *smooth*, and 0 otherwise. During the variable fixing stage, additional measures may be taken so that the sub-MIP remains feasible. Specifically for the CLSP problem, it is prevented that more than one setup is performed to/from any given product. Figure 4.1 illustrates the combination of the exponential smoothing with the constructive mask to build a sub-MIP.

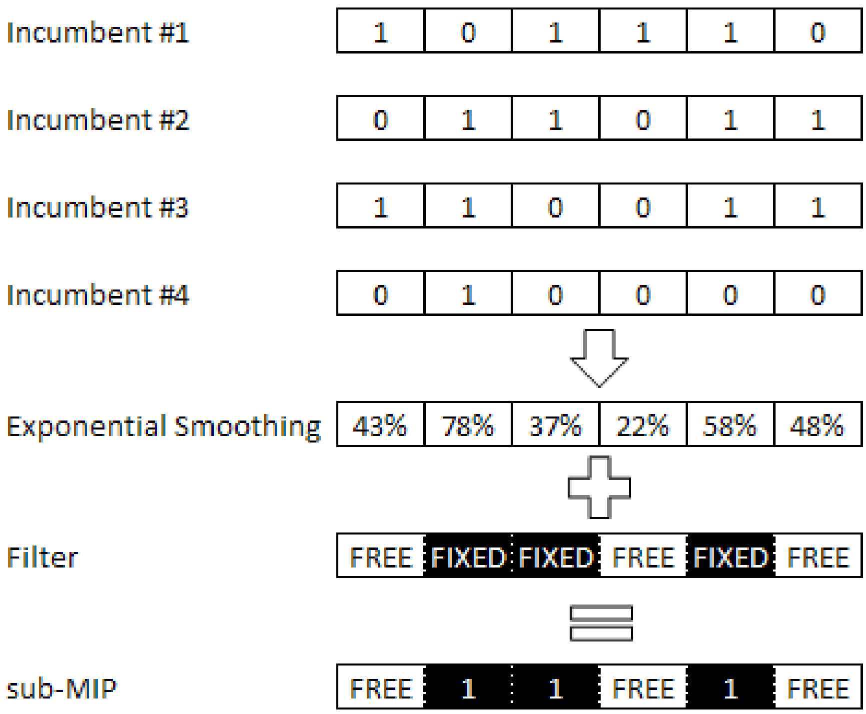


Figure 4.1: Pattern information tracked by an exponential smoothing variable combines with a filter mask to create a sub-MIP

The sub MIP is then solved. While the sub MIP is greatly simpler than the original MIP, it may still require an enormous amount of time to solve to optimality. Therefore, a stopping criterion must be set. During our preliminary tests, the number of nodes explored in the sub-MIP tree was limited according to the sub-MIP size.

Depending on the obtained results, the f factor is dynamically changed, so as to balance speed in finding new incumbents (smaller, more constrained sub-MIPs) and the feasibility of the generated models (bigger, less constrained sub-MIPs). If the sub-MIP returns a feasible integer solution that is better than the previous incumbent candidate, we increase f and inject the solution into the original branch-and-cut tree. Otherwise, we decrease f and resume the original branch-and-cut algorithm until a new incumbent candidate is found. Either way, the pattern recognition method is applied to the new incumbent, and the entire cycle is repeated until the branch-and-cut tree assures the optimal solution was found, ending the method. Figure 4.2 represents the overall cyclic structure of the hybrid metaheuristic proposed.

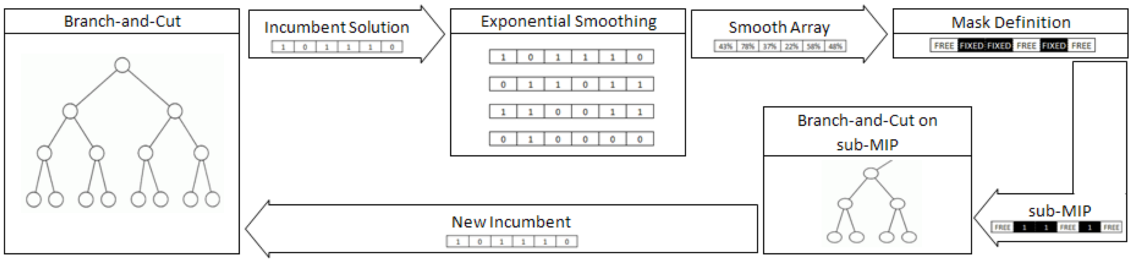


Figure 4.2: Hybrid Metaheuristic Diagram

Preliminary tests suggest that the method quickly improves the first incumbent candidates provided by CPLEX during its branch-and-cut method. This is useful not only because good solutions are quickly found, but also because upper bounds are quickly set, thus pruning branches and decreasing the size of the explored tree. It is also suggested that solutions found by the method are rarely found by CPLEX without spending a great amount of time exploring the branch tree, which may be due to "hidden" local optima that are not evident to the optimization engine when analyzing the original MIP, but in turn are quickly explored (and therefore found) in the sub-MIP. However, the method rapidly dries out, and later attempts hardly produce better results. Overall, preliminary tests were not satisfactory when matched against a standard branch-and-cut approach, raising several hypothesis.

The first hypothesis is that the parameters used were not optimal. Specifically, the smoothing factor and the sub-MIP tree node limit have a great impact on the overall performance of our method. During our preliminary tests, no combination was found that consistently produced satisfactory results. However, the hypothesis of finding a good combination is not discarded, and can be further explored.

A second hypothesis is that the probability for a certain variable to get fixed or not in the sub-MIP problem should depend on whether or not a strong pattern was identified for it. This approach would represent a closer take on the second assumption, meaning that when a pattern is found, the chances of the corresponding variable getting fixed in the sub-MIP should be improved.

A third hypothesis is that the method should use a better starting value. Using a constructive heuristics to provide a starting point would impose an upper limit to the branch-and-cut tree, thus reducing its size, while at the same time providing a good initial feed to the hybrid metaheuristic, putting focus on polishing it rather than using it for pattern recognition, which would represent a closer take on the first assumption.

While results turned out to be unsatisfactory, this project has provided the authors with useful insights and techniques that may be useful for future research. Much progress was made in understanding the mechanisms of interaction between heuristic and exact methods, specifically on linking programming tools developed in C++ to a commercial optimization engine, ILOG CPLEX, opening ways for the development, implementation and testing of new hybrid methods.

As a key learning, we believe that the balance between exact and heuristic methods must be well thought. It assumes a vital role in the performance of hybrid methods, specially those that rely on successive iterations between the two components. Ideally, this balance should be dynamically adjusted, allowing the hybrid method to adapt and react to different stages of the solution process.

Chapter 5

Conclusion

In this work, we have presented a novel formulation for CLSP which correctly handles non-triangular setup costs and times while enforcing the necessary feature of minimum lot size, and allows setup cross overs between adjacent periods. This extensions open ways for solutions not being considered so far, which may lead to improvements in overall production planning efficiency, and reduce the total cost of production plans.

Additionally, we have developed a method for dynamically identifying and removing disconnected sub tours. Such a method is required for large problems, since the direct implementation of our model would require the use of an exponential number of constraints. The simplicity of this method makes it easy to implement in most programming languages usually combined with optimization engines. The algorithms were implemented in ILOG/CPLEX through C++, and are available on request. Nevertheless, an important future research question is to find a polynomial sized set of constraints that cut disconnected sub tours off, while enabling all types of connected cycles.

Extension to this model considering multiple machines, shortages, backlogging costs and maximum lot sizes are straightforward, making it a good starting point for models reflecting a wide range of real life situations.

The two proposed modelling extensions for the capacitated lotsizing and scheduling problem have originated an article, submitted to a scientific journal and currently awaiting for revision.

We have also explored a hybrid metaheuristic that attempts to improve the quality of solutions provided by the branch-and-cut algorithm by detecting good solution patterns and, based on those, applying a large neighborhood search framework. Preliminary results are unsatisfactory, as the branch-and-cut algorithm consistently beats the proposed hybrid metaheuristic. However, valuable insights have been extracted from this experience, and ways of achieving efficient hybrid interaction between a programming language (C++) and a commercial optimization engine (ILOG CPLEX) were explored, providing useful information for future research projects, whose focus certainly lies on developing robust hybrid metaheuristic approaches to the CLSP.

Bibliography

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.

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.

E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90, 2005.

S. Fernandes and H. Lourenço. A grasp and branch-and-bound metaheuristic for the job-shop scheduling. *Lecture Notes in Computer Science - Springer Berlin / Heidelberg*, 4446: 60–71, 2007.

M. Fischetti and A. Lodi. Repairing mip infeasibility through local branching. *Computers & Operations Research*, 35(5):1436–1445, 2008.

J. M. Framinan and R. Pastor. A proposal for a hybrid meta-strategy for combinatorial optimization problems. *Journal of Heuristics*, 14(4):375–390, 2008.

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

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

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):159–169, 2000.

A. A. Menezes, B. Almada-Lobo, and A. Clark. Capacitated lotsizing and scheduling with sequence-dependent, period overlapping and non triangular setups. *Submitted to Journal of Scheduling*, 2008.

P. Porkka and A. P. J. V. M. Kuula. A modified framework for modelling set-up carryover in the capacitated lotsizing problem. *International Journal of Production Research*, 38(14): 3421–3424, 2000.

E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *Informs Journal on Computing*, 19(4):534–541, 2007.

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

C. Suerie. Modeling of period overlapping setup times. *European Journal of Operational Research*, 174(2):874–886, 2006.

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

C. Sung and C. T. Maravelias. A mixed-integer programming formulation for the general capacitated lot-sizing problem. *Computers & Chemical Engineering*, 32:244–259, 2008.

E. A. V. Toso and R. Morabito. Otimização no dimensionamento e seqüenciamento de lotes de produção: Estudo de caso numa fábrica de rações. *Gestão e Produção*, 12(2):203–217, 2005.