UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Diego Jacinto Fiorotto

Modelos Matemáticos e Métodos de Solução para Problemas de Dimensionamento de Lotes

Tese de Doutorado

São José do Rio Preto

Março de 2015

Diego Jacinto Fiorotto

Modelos Matemáticos e Métodos de Solução para Problemas de Dimensionamento de Lotes

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Silvio Alexandre de Araujo

São José do Rio Preto

Março de 2015

Fiorotto, Diego Jacinto.
Modelos matemáticos e métodos de solução para problemas de dimensionamento de lotes / Diego Jacinto Fiorotto. -- São José do Rio Preto, 2015
111 f. : il., tabs.

Orientador: Silvio Alexandre de Araujo
Tese (doutorado) – Universidade Estadual Paulista “Júlio de Mesquita Filho”, Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Modelos matemáticos. 3. Problema de dimensionamento de lotes. 4. Planejamento experimental. I. Araujo, Silvio Alexandre de. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 518.721

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Diego Jacinto Fiorotto

Modelos Matemáticos e Métodos de Solução para Problemas de Dimensionamento de Lotes

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Banca Examinadora

Prof. Dr. Silvio Alexandre de Araujo
UNESP - São José do Rio Preto
Orientador

Profa. Dra. Maria do Socorro Nogueira Rangel
UNESP - São José do Rio Preto

Prof. Dr. Pedro Munari
UFSCAR - Universidade Federal de São Carlos

Prof. Dr. Mauricio Cardoso de Souza
UFMG - Universidade Federal de Minas Gerais

Prof. Dr. Edson Luiz França Senne
UNESP - Guaratinguetá

São José do Rio Preto,
Março de 2015.

Aos meus amados pais, Dirceu e Ivone,
que lutaram por mim em todos os sentidos;
À minha amada irmã, Samira,
pelo apoio incondicional;
À minha namorada, Camila,
por todo carinho e amor.
Dedico.

Agradecimentos

Chega o fim de mais uma etapa da minha vida, uma grande conquista, e por mais que eu me esforçasse jamais conseguiria colocar aqui uma lista completa das pessoas que me ajudaram, direta ou indiretamente, para a obtenção deste título. Fica aqui meu profundo agradecimento a todos. Em especial agradeço:

À Deus, por todas as oportunidades oferecidas, pela sabedoria para superar os momentos difíceis e pelas inúmeras felicidades que vivi durante a realização deste trabalho, nos quais tive o prazer de conhecer pessoas e lugares que serão lembrados para sempre.

Aos meus pais, Dirceu e Ivone, minha eterna gratidão pelo incentivo, confiança e por todo sacrifício realizado para que eu pudesse sempre me dedicar apenas aos estudos.

À minha irmã Samira, pela cumplicidade e amor incondicional dedicado desde sempre.

Agradeço especialmente ao meu orientador, Silvio, por todos os ensinamentos, amizade, dedicação e paciência durante todos estes anos que trabalhamos juntos, desde as orientações de Iniciação Científica. Muito obrigado por todas as oportunidades oferecidas e pela confiança à mim depositada desde sempre.

Ao Prof. Raf Jans, pela valiosa contribuição para a realização deste trabalho. Agradeço pela receptividade, pela paciência com o idioma e por todos os ensinamentos durante o período que fiquei em Montreal.

À minha namorada, Camila, que sempre esteve ao meu lado nos momentos mais difíceis, pelo carinho, companheirismo e por sua compreensão nos vários momentos em que precisei estar ausente para a realização deste trabalho.

Aos amigos Tiago, Gislaine, Michelli e Rafael, pelas várias ajudas e por tornarem as horas de estudos na salinha da pós muito mais agradáveis.

À todas as pessoas que conheci em Montreal em especial ao André, Rodrigo, Dauwe, Michelle, Gianluca, Serena, Breno, Sofiene e todos os outros amigos do CIRRELT que foram minha família no Canadá e fizeram com que esta experiência fosse uma das melhores da minha vida.

Aos professores de graduação e pós-graduação, pelos valiosos ensinamentos e por ajudarem no meu crescimento intelectual e profissional.

À todos os colegas, pessoas e funcionários do IBILCE que, direta ou indiretamente, contribuíram para a elaboração deste trabalho. Em especial aos funcionários do PPGMAT sempre muito atenciosos e competentes.

À FAPESP, pelo apoio financeiro.

Resumo

O problema de dimensionamento de lotes é um problema de otimização da produção e consiste em determinar a quantidade de produtos a serem produzidos em cada período ao longo de um horizonte de tempo finito, de modo a atender uma demanda e otimizar uma função objetivo, por exemplo, minimizar os custos. Esta tese aborda duas extensões diferentes do problema de dimensionamento de lotes padrão. Na primeira parte, considera-se o problema de dimensionamento de lotes com vários itens, tempos de preparação e máquinas paralelas distintas, e na segunda parte, o problema de dimensionamento de lotes com vários itens e preparação *crossover*. Para a primeira parte desta tese, em que estuda-se o problema de dimensionamento de lotes com máquina paralelas, o objetivo é aplicar diferentes métodos de solução que utilizam relaxação Lagrangiana e decomposição de Dantzig-Wolfe para obter limitantes inferiores de alta qualidade e desenvolver heurísticas Lagrangianas para obter boas soluções factíveis (limitantes superiores). Baseado em uma reformulação forte do problema como um problema de caminho mínimo e diferente da abordagem tradicional em que as restrições de ligação são as restrições de capacidade, utiliza-se as restrições de fluxo, isto é as restrições de demanda, como as restrições de ligação. O objetivo desta abordagem é obter limitantes inferiores de alta qualidade e para tanto, utiliza-se três métodos de solução diferentes. No primeiro a relaxação Lagrangiana é aplicada as restrições de fluxo. Para os outros dois resolve-se o problema mestre aplicando métodos de solução diferentes que combinam relaxação Lagrangiana e decomposição de Dantzig-Wolfe de forma hibrida. Duas heurísticas primais, baseadas em transferências de produção, são utilizadas para gerar soluções factíveis. Experimentos computacionais utilizando conjuntos de dados da literatura são apresentados e mostram que os métodos de solução produzem limitantes inferiores de excelente qualidade e limitantes superiores competitivos, quando comparados com os limitantes produzidos por outros métodos da literatura e por um pacote computacional de alta performance. Na segunda parte da tese, estuda-se o problema de dimensionamento de lotes com preparação *crossover*, em que a primeira operação de preparação de cada período de planejamento pode ser inicializada no período anterior, se não foi utilizada toda capacidade naquele período. Isto fornece mais flexibilidade no planejamento e aumenta a possibilidade de encontrar mais soluções factíveis e melhores soluções se comparado com a suposição padrão. Duas formulações diferentes apresentadas na literatura para modelar a preparação *crossover* foram analisadas. Dado que estas formulações não foram comparadas diretamente, apresenta-se um estudo computacional para determinar qual é a melhor formulação. Além disso, explora-se ideias para evitar a definição de variáveis binárias extras para modelar a preparação *crossover* e propõem-se restrições de quebra de simetrias para ambas formulações da literatura. Por fim, quantifica-se o valor desta flexibilidade com um experimento computacional e analisa-se quais fatores influenciam este valor.

Palavras-chave: *Dimensionamento de Lotes; Máquinas Paralelas; Preparação Crossover; Métodos Híbridos; Modelos Matemáticos.*

Abstract

The lot sizing problem is a production optimization problem and consists of determining the quantity of products to be produced in each period of a finite time horizon, in order to meet the demand and optimize an objective function, for example, to minimize costs. In this thesis we address two different extensions of the standard lot sizing problem. In the first part we consider the capacitated lot-sizing problem with multiple items, setup time and unrelated parallel machines and, in the second one, the capacitated lot sizing problem with multiple items and setup crossover. For the first part of this thesis where we study the lot sizing problem with unrelated parallel machines, the aim is to apply different solution methods that use Lagrangian relaxation and Dantzig-Wolfe decomposition to obtain high quality lower bounds and develop Lagrangian heuristics to obtain good feasible solutions (upper bounds). Based on a strong reformulation of the problem as a shortest path problem and unlike in the traditional approach in which the linking constraints are the capacity constraints, we use the flow constraints, i.e. the demand constraints, as linking constraints. The aim of this approach is to obtain high quality lower bounds and for this we have used three different solution methods. In the first one the Lagrangian relaxation is applied to the flow constraints. For the other two we solve the master problem applying solution methods that combine Lagrangian relaxation and Dantzig-Wolfe decomposition in a hybrid form. Two primal heuristics, based on transfers of production quantities, are used to generate feasible solutions. Computational experiments using data sets from the literature are presented and show that the solution methods produce lower bounds of excellent quality and competitive upper bounds, when compared with the bounds produced by other methods from the literature and by a high-performance MIP software. Next, in the second part of the thesis, we look at the lot sizing problem with setup crossover, in which the first setup operation of each planning period can already start in the previous period, if not all the capacity is used in that previous period. This provides more flexibility in the planning and increases the possibility of finding feasible and better solutions compared to the standard assumption. Two different formulations presented in the literature to model the setup crossover were analysed. Since these formulations have not been compared directly to each other, we present a computational study to determine which is the best formulation. Furthermore, we explore ideas to avoid the necessity of defining new extra binary variables to model the setup crossover and propose symmetry breaking constraints for both formulations from the literature. Finally, we quantify the value of this type of flexibility in a computational experiment and analyse which factors influence this value.

Keywords: Lot Sizing; Parallel Machines; Setup Crossover; Hybrid Methods; Mathematical Models.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Diego Jacinto Fiorotto

Mathematical Models and Solution Methods for Lot Sizing Problems

Tese de Doutorado

São José do Rio Preto

Janeiro de 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is a wide variety of models for production planning and inventory management. This thesis addresses problems that arise in the context of the industrial production plan. In general, the production planning considers decisions of acquiring resources and raw materials, as well as, decisions about production activities necessary to transform raw materials into final items in an efficient way. In some industry, such decisions about production activities are narrowly related to decisions about the lot sizes of final items that must be produced. In this context arises the lot sizing problem which consists, basically, of determining the size of the production lots (i.e., the amounts of each item to be produced) in each of the periods in the planning horizon, in order to minimize the production and inventory costs (or maximize profits), satisfy constraints of resources available and meet the demand of items. In this text, we consider a finite time horizon subdivided into periods, and the demand for each item in each period is known and dynamic, i.e., it varies over the planning horizon.

The lot-sizing problems have been studied for over 50 years. The crucial decision to be made is related to the ideal balance between inventory and setup for the machines. Generally, when one decides to start production of a particular item, there is a fixed setup for the machine being used, the goal is to make decisions about the amount to be produced according to demand. The production of a small lot to meet only immediate demands will result in a low inventory cost. However, a new setup of the machine will have to be paid in the near future of the planning horizon. On the other hand, the production of a large amount, not only to meet immediate demands, will result in high inventory costs, but it will avoid the necessity of a new setup in a short time.

With the natural evolution of the industrial decisions process due, among other factors, the increased competitiveness imposed by the current global market, several strategies have been used to improve industrial decisions, making them more complex. Therefore, there has been an increasing interest in the application of mathematical models to solve

industrial problems, and researchers have been able to incorporate more and more real world features into lot sizing problems. This thesis aligns with this trend of evolution in the decision process.

The first problem studied in this thesis involves the production planning of multiple items in a single stage. The production sector is composed of unrelated parallel machines with limited capacity. The items can be produced on any of the machines and to start producing an item incurs a setup time for the machine being used. In practical production planning problems, parallel machines often need to be taken into account.

The production planning in industries that consider several items and multiple machines is a complex task and must be routinely performed. Note that the combinatorial nature of such problems make them difficult to solve and it requires the development of complex mathematical/computational tools. Among the studies that consider a production environment with parallel machines, some of them have taken into account the setup times of the machines, which increases significantly the complexity of the problems. In this thesis, we have studied the lot sizing problem on parallel machines considering the setup times of the machines.

The aim of this first part of the thesis is to develop solution methods for the lot sizing problem on parallel machines. To get good lower bounds (dual bounds) for the problem, we have used a strong reformulation of the problem based on the shortest path problem, and proposed solution methods that use Lagrangian relaxation and Dantzig-Wolfe decomposition, so that unlike most studies that use the usual decomposition, i.e., for items (by relaxing the capacity constraints), we use the periods and machines decomposition (by relaxing the flow constraints).

The Dantzig-Wolfe decomposition and Lagrangian relaxation are important techniques in the context of solving integer programming problems. We present methods that use these two techniques separately and in a hybrid way with a focus on the discussion of how the relationship between them can be exploited for developing algorithms combining the strengths of both techniques. We highlight the importance of studying and get good lower bounds for these problems due to the computational complexity that can reduce considerably the solution trees of the exact algorithms.

For the upper bounds (primal bounds) we have developed two Lagrangian heuristics (the second one is an extension of the first one) based on transfers of production. The idea is that all production plans from the subproblems arising from Lagrangian relaxation and Dantzig-Wolfe decomposition, respect the capacity constraints (these constraints are in the subproblems). Therefore, the search for feasible solutions lies in meeting the demands, given that the flow constraints were relaxed.

The second problem studied in this thesis involves the production planning of multiple

items allowing the possibility of setup crossover. In the standard version of the lot sizing problem (Trigeiro et al., 1989) the setup for the first product type produced in a period starts at the beginning of that period. In this part of the thesis, we study an extension of the standard assumption that includes the possibility of a setup crossover. The idea is that in certain cases setup operations can be interrupted at the end of a period and resumed at the beginning of the next period. In other words, the setups can span over two periods. This implies that the first setup in period t can already start at the end of period $t-1$ if there is some capacity left, and continue at the beginning of period t.

Although setup crossover is a natural extension of the standard assumption, just a few studies have considered it, due to the difficulty in dealing with the underlying problems (Mohan et al., 2012; Belo-Filho et al., 2014). All the studies that handle setup crossovers in their formulations have added extra binary variables to the formulations indicating if there is a setup crossover in a period or not, which increases the difficulty of the formulations. The aim of this part is mainly to propose new ideas to avoid the necessity of defining new extra binary variables to model the setup crossover, to propose new constraints to break the symmetry which is present in the formulations from the literature and analyse the impact of the proposed adaptations of these formulations.

The remaining chapters of this thesis is organized as follows. In Chapter 2, we provide a study on lot sizing problems with unrelated parallel machines. A reformulation of the problem is proposed using the strategy of variable redefinition proposed by Eppen and Martin (1987), and a Lagrangian heuristic is developed in which the demand constraints are relaxed, thus the problem is decomposed per period and per machine instead of the classical per item decomposition. In Chapter 3 we keep studying the lot sizing problem with unrelated parallel machines and present two hybrid solution methods to find lower bounds that combine Lagrangian relaxation and Dantzig-Wolfe decomposition so that in both methods the Dantzig-Wolfe decomposition is applied using as linking constraints the flow constraints. In the first hybrid method the main idea is instead of using the simplex algorithm to solve the restricted master problem in order to obtain the optimal dual values of the extended formulation (master problem) it is possible to use the Lagrangian relaxation applied to this formulation to approximate these values. For the second one the idea is to use the Lagrangian relaxation in the extended formulation (master problem) to approximate the dual variables (as in the previous method), and afterwards use the Lagrangian relaxation also to generate columns to the restricted master problem. Furthermore, we extend the feasibility heuristic proposed in Chapter 2. Chapter 4 deals with the lot sizing problem with setup crossover. A reformulation of the problem using the simple plant location model is used, three new formulations avoiding the necessity to define new extra binary variables to model the setup crossover and two new symmetry

breaking constraints are proposed. Finally, we provide a substantial computational study about the value of the possibility of setup crossover considering problems with different characteristics. Our conclusions and ideas for future research are presented in Chapter 5.

Chapter 2

Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines

In this chapter we consider the capacitated lot sizing problem with multiple items, setup time and unrelated parallel machines.[1] The items can be produced on any of the machines and several items can be produced on the same machine in the same time period (large bucket model). At the start of the production of each item, there is a setup time for the machine used and the setup is not sequence-dependent.

The aim of this chapter is to find good lower and upper bounds for the single stage problem with unrelated parallel machines and, for this, a reformulation based on the shortest path problem is proposed and the bounds are generated with a Lagrangian heuristic. Different from most other papers that use the traditional decomposition method (i.e., per item), decomposition per period and machine is used here. Therefore, the first contribution of this chapter is to generalize the reformulation proposed in Jans and Degraeve (2004a), that considers the single machine, to the case of unrelated parallel machines. Secondly, to develop a period and machine decomposition for this problem and use Lagrangian relaxation to obtain high quality lower bounds for the parallel machine problem. As a third contribution, we develop an innovative primal heuristic, based on transfer of production, to make feasible the demand constraints and finally we perform a computational study in which we compare to another method in literature (Toledo and Armentano, 2006), which is based on a different type of decomposition, and to the computational package CPLEX.

[1] A compact version of this chapter has been published in the *Annals of Operations Research* (see Fiorotto and Araujo, 2014).

2.1 Literature review

In this section, we will first discuss papers related to lot sizing problems on parallel machines and subsequently we discuss relevant papers involving reformulations for lot sizing problems.

2.1.1 Literature review for lot sizing on parallel machine

In practical production planning problems, parallel machines often need to be taken into account. Areas of production that consider parallel machines are, for example, the pharmaceutical industry (De Matta and Guignard, 1995), plastic sheet production (Mergaux and van Wassenhove, 1984), tile production (De Matta and Guignard, 1994), the tire industry (Jans and Degraeve, 2004b), bottling of liquids and others (Carreno, 1990) and packaging (Marinelli et al., 2007).

Considering the problem with identical parallel machines, Lasdon and Terjung (1971) propose a heuristic for a lot sizing and scheduling problem with no machine setup time. Sung (1986) proposes a dynamic programming algorithm for the single-item problem without capacity constraints. Carreno (1990) proposes a heuristic for the Economic Lot Scheduling Problem (ELSP) i.e., with a constant demand rate, with setup times for parallel machines and solves problems with one hundred items and ten machines in fast computational times. Jans (2009) proposes new constraints to break the symmetry that is present due to the identical machines and tests his approach using a network reformulation for the problem. Tempelmeier and Buschkuhl (2009) consider the multi-stage problem with setup carry-over (a setup is maintained between adjacent periods) and develop a Lagrangian heuristic.

For the unrelated parallel machines case, Toledo and Armentano (2006) relax the capacity constraints and proposes a Lagrangian heuristic to solve the problem. An initial solution is obtained by minimizing the Lagrangian problem, which is normally infeasible. In an attempt to make it feasible, production is shifted between periods and machines, moving the production that exceeds the capacity and looking for feasible solutions that minimize the cost. In her PhD thesis, Toledo (1998) presents two branch-and-bound algorithms to solve this problem exactly. The first algorithm is based on a mixed-integer programming formulation and Lagrangian relaxation of the capacity constraints. The second was developed from the representation of the problem as a generalized network and linear relaxation. The two exact algorithms are used only to solve small instances. Fiorotto and Araujo (2012) study the same problem. The authors use a strong reformulation of the problem and instead of the capacity constraints, they relax the demand constraints using Lagrangian relaxation and obtain better lower bounds.

Multi-stage problems with unrelated parallel machines were studied in Ozdamar and Birbil (1998), who present a generic model in which the multi-stage case can be considered. As well as its regular capacity, each machine has permission to use a fixed quantity of overtime for a given cost. The model does not allow for the division of lots, i.e. a total quantity of production of an item per period must be produced on a single machine, therefore, only one of the parallel machines can be used to produce a given specific item per period. Three similar hybrid heuristics are developed to solve the problem, in which a Tabu search algorithm is used to find feasible solutions and to improve the solutions. Problems with up to twenty items, six periods and five machines are solved. Stadtler (2003) and Helber and Sahling (2010) also analyze the multi-stage problem. Stadtler (2003) proposes a period decomposition heuristic and, to solve each subproblem, a reformulation based on the facility location problem is used. Helber and Sahling (2010) propose a fix-and-optimize approach and obtain better results than those obtained by Stadtler (2003).

Some research in the literature deals with the problem of parallel machines and sequence-dependent setup costs and times. Some of the research papers consider small bucket models, where only one item can be produced per machine per period. Salomon et al. (1991) study the Discrete Lot Sizing and Scheduling Problem (DLSP) with parallel machines, and analyze the complexity for the cases of identical and non-identical machines. The authors also present some solution algorithms. Kang et al. (1999) propose a method based on column generation and branch-and-bound. In addition, they adapt the method to solve some real problems heuristically. Belvaux and Wolsey (2000) describe a generic model and an optimization system that is capable of solving a wide range of lot sizing problems including special cases with different items, parallel machines and various time periods. The problems are solved using routines from the optimization software package XPRESS-MP. In addition, a primal heuristic can be integrated into the solution process. The authors solve a large number of instances taken from the literature. Meyr (2002) present a general model that consists of an extension of the General Lot Sizing and Scheduling Problem (GLSP) model for the case in which both setup cost and time are sequence-dependent. A new solution method is presented combining a dual re-optimization method with local search techniques to fix the binary variables that represent the setup sequence. For each fixed sequence, a network flow algorithm with dual re-optimization is solved and the solution value is used to evaluate the quality of the current solution. According to the authors, the computational results demonstrate the efficiency of the new method. Fandel and Stammen-Hegener (2006) also present a model based on the GLSP model and consider the multi-stage case. The model assumes that items are produced on different machines with the general production structure. Each

machine has its own production capacity and their production costs can vary per period. Each item can not be produced in more than one micro-period per macro-period. Due to the model's complexity, only instances with a small number of items and macro-periods can be optimally solved. Marinelli et al. (2007) proposes a solution approach for a real capacitated lot sizing and scheduling problem with parallel machines and shared buffers, arising in a packaging company producing yoghurt. A new effective two stage optimization heuristic based on the decomposition of the problem into a lot sizing problem and a scheduling problem has been developed. Finally, Meyr and Mann (2013) propose a heuristic for the lot sizing and scheduling problem on parallel machines. Different decomposition approaches are proposed and compared with results from the literature.

2.1.2 Literature review on reformulations for lot sizing problems

Considering reformulation for lot sizing problems recently, various research papers (Chen and Thizy, 1990; Diaby et al., 1992; Alfieri et al., 2002; Pochet and van Vyve, 2004; Jans and Degraeve, 2004a; Denizel and Süral, 2006; Denizel et al., 2008; Jans, 2009; Süral et al., 2009) have used alternative formulations to the classical formulation (see Sections 2.2.2 and 4.3.1) of the lot sizing problem, because the linear relaxations of these alternative formulations are stronger in the sense that they reach better lower bounds which benefits some solution methods, for instance those that use the objective function of a linear relaxation as lower bounds for the optimal solution, and also those that use rounding procedures on the relaxed solution.

Within such alternative formulations, two of them deserve more attention. The first one deals with the reformulation of the problem as a Shortest Path problem (SP) in which a redefinition of the variables proposed by Eppen and Martin (1987) is the strategy used. The second one consists of a reformulation based on the Facility Location problem (FL) studied in Krarup and Bilde (1977).

Various theoretical and computational results concerning such reformulations have been published in the literature. Nemhauser and Wolsey (1988) prove that, considering the single-item, single machine problem and capacity constraints the SP and FL formulations are equivalent and that the SP and FL linear relaxations describe the convex hull of the problem. Brahimi et al. (2006) present a revision including different formulations for this problem.

Thizy and van Wassenhove (1985) and Alfieri et al. (2002) apply such reformulations to the problem with multiple items, single machine, capacity constraints and without setup times, and the computational tests presented in Alfieri et al. (2002) show that the value of the objective function of the linearly relaxed SP and FL models are 60% higher than those of the classical formulation.

For the same problem with setup time but without setup cost, Denizel and Süral (2006) also compare the value of the objective function of the linearly relaxed SP and FL models with those of the classical formulation and conclude that SP and FL models are 30% higher. Süral et al. (2009) consider an alternative formulation and use Lagrangian relaxation to solve the problem and conclude that their method on average provides about 21% improvement in the duality gap over its best known competitor.

Considering the problem with multiple items, single machine, capacity constraints, setup cost and setup times, Diaby et al. (1992) use Lagrangian relaxation in conjunction with a reformulation of the problem as a transportation problem. Jans and Degraeve (2004a) present a method to get lower bounds for the problem considering a formulation based on the shortest path problem. Denizel et al. (2008) prove the equivalence between the linear relaxations of the SP and FL reformulations. Araujo et al. (2015) develop a transformed reformulation and design a fast heuristic procedure for this problem that provides good solutions and a strong lower bound used to assess the solution quality.

Jans (2009) applies the reformulation SP for the problem considering identical parallel machines and proposes new constraints to break the symmetry of this problem. For the problem with unrelated parallel machines Fiorotto and Araujo (2012) extend the ideas proposed by Jans (2009) to reformulate the problem as a shortest path problem and apply the Lagrangian relaxation to find good lower bounds.

Finally, we observe that in addition to the papers already mentioned, several authors have proposed these reformulations for other extensions of the lot sizing problem, for example, Araujo and Clark (2013) for the lot sizing problem with sequence-dependent, Belo-Filho et al. (2014) for the lot sizing problem with setup carryover and crossover and Akartunali and Miller (2012) for the multi-level problem.

2.2 Problem formulations

2.2.1 Classical formulation

We first present the classical formulation of the lot sizing problem on unrelated parallel machines. This formulation is based on the formulation of Trigeiro et al. (1989) for the single machine problem, and has been studied in Toledo and Armentano (2006).

For the mathematical formulation of the problem consider the following data:

$I = \{1, ..., n\}$: set of items;

$J = \{1, ..., r\}$: set of machines;

$T = \{1, ..., m\}$: set of periods;

d_{it}: demand of item i in period t;

$sd_{it\tau}$: the sum of the demand for item i, from period t until period τ ($\tau \geq t$);
hc_{it}: unit inventory cost of item i in period t;
sc_{ijt}: setup cost for item i on machine j in period t;
vc_{ijt}: production cost of item i on machine j in period t;
fc_i: unit cost of initial inventory for item i;
st_{ijt}: setup time for item i on machine j in period t;
vt_{ijt}: production time of item i on machine j in period t;
Cap_{jt}: capacity (in units of time) of machine j in period t.

The decision variables are then defined as follows:
x_{ijt}: number of units produced of item i on machine j in period t;
y_{ijt}: binary variable, indicating the set up or not of item i on machine j in period t;
s_{it}: quantity of inventory of item i at the end of period t;
s_{i0}: initial quantity of inventory for item i.

Mathematical formulation:

$$Min \sum_{i=1}^{n} fc_i s_{i0} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}(sc_{ijt}y_{ijt} + vc_{ijt}x_{ijt}) + \sum_{i=1}^{n}\sum_{t=1}^{m} hc_{it}s_{it} \tag{2.1}$$

Subject to:

$$s_{i,t-1} + \sum_{j=1}^{r} x_{ijt} = d_{it} + s_{it} \qquad \forall i \in I, t \in T \tag{2.2}$$

$$x_{ijt} \leq min\{(Cap_{jt} - st_{ijt})/vt_{ijt}, sd_{itm}\}y_{ijt} \qquad \forall i \in I, j \in J, t \in T \tag{2.3}$$

$$\sum_{i=1}^{n}(st_{ijt}y_{ijt} + vt_{ijt}x_{ijt}) \leq Cap_{jt} \qquad \forall j \in J, t \in T \tag{2.4}$$

$$y_{ijt} \in \{0,1\},\ x_{ijt} \geq 0,\ s_{it} \geq 0,\ s_{i0} \geq 0,\ s_{im} = 0 \quad \forall i \in I, j \in J, t \in T \tag{2.5}$$

The objective function (2.1) minimizes the total setup, production, inventory and initial inventory costs. The constraints (2.2) guarantee the inventory balance in each period. To avoid infeasible problems, the model considers the possibility of initial inventory. However the cost fc_i for this inventory is very large, thus s_{i0} will have a value of zero, only becoming different (non-zero) when a feasible solution is not found. Next are the machine setup constraints (2.3) and the capacity limits (2.4). In order to make the formulation stronger, we limit the production for each item in constraints (2.3) by both the remaining demand and the maximum possible production with the available capacity. Finally, constraints (2.5) define the variables domains.

2.2.2 Reformulation as a shortest path problem

Next we present a reformulation of the model (2.1)-(2.5) using the variable redefinition approach proposed by Eppen and Martin (1987), producing a formulation based on the shortest path problem. We observe that Eppen and Martin (1987) have considered the problem with a single machine without capacity constraints. For this formulation, each node on the graph represents a period, including a dummy period $m+1$. There is an arc between each pair of nodes and the arc between nodes t and q ($q > t$) represents the option of producing the sum of the demands between period t and period $q-1$ during period t. The cost of each arc corresponds to the total production and inventory cost associated with the variable. The objective is to find the shortest path from 1 to $m+1$. Note that with this definition the problem can be interpreted as a multicommodity network flow problem.

For the reformulation the following parameters are defined:

cv_{ijtk}: cost of production and inventory holding to produce item i on machine j in period t meeting the demand for periods t to k:

$$cv_{ijtk} = vc_{ijt}sd_{itk} + \sum_{s=t+1}^{k}\sum_{u=t}^{s-1} hc_{iu}d_{is};$$

ci_{it}: cost of initial inventory of item i meeting demand for the periods 1 through to period t:

$$ci_{it} = fc_i sd_{i1t} + \sum_{s=2}^{t}\sum_{u=1}^{s-1} hc_{iu}d_{is}.$$

There are also the following new variables for the model:

zv_{ijtk}: fraction of the production plan for item i on machine j, in which the production in period t meets the demand for the period t to period k;

w_{it}: fraction of the initial inventory plan for item i in which the demand is met for the first t periods.

The reformulation based on the shortest path problem is as follows:

$$Min \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m} sc_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}\sum_{k=t}^{m} cv_{ijtk}zv_{ijtk} \tag{2.6}$$

Subject to :

$$1 = \sum_{k=1}^{m} w_{ik} + \sum_{j=1}^{r}\sum_{k=1}^{m} zv_{ij1k} \qquad \forall i \in I \tag{2.7}$$

$$w_{i,t-1} + \sum_{j=1}^{r}\sum_{k=1}^{t-1} zv_{ijk,t-1} = \sum_{j=1}^{r}\sum_{k=t}^{m} zv_{ijtk} \qquad \forall i \in I, t \in T \setminus \{1\} \tag{2.8}$$

$$\sum_{k=t}^{m} zv_{ijtk} \leq y_{ijt} \qquad \forall i \in I, j \in J, t \in T \qquad (2.9)$$

$$\sum_{i=1}^{n} st_{ijt} y_{ijt} + \sum_{i=1}^{n} \sum_{k=t}^{m} vt_{ijt} sd_{itk} zv_{ijtk} \leq Cap_{jt} \qquad \forall j \in J, t \in T \qquad (2.10)$$

$$y_{ijt} \in \{0,1\}, \quad w_{it} \geq 0 \qquad \forall i \in I, j \in J, t \in T \qquad (2.11)$$

$$zv_{ijtk} \geq 0 \qquad \forall i \in I, j \in J, t \in T \ \forall k \in T, k \geq t \qquad (2.12)$$

The objective function (2.6) minimizes the sum of the setup, production and inventory costs including initial inventory costs. The constraints (2.7) and (2.8) define the flow balance constraints for the shortest path network. For each item, one flow unit is sent through the network, imposing that the demand for each product is met on time. Constraints (2.9) are the setup forcing constraints. The capacity constraints (2.10) limit the total setup and production times to the available capacity in each period and for each machine. Constraints (2.11) and (2.12) define the variable domains.

Figure 2.1 represents the network constraints for a three period problem for one item and specific machine. The flow constraint for the first period (2.7) is represented by Figure 2.2: a flow of one unit is coming into node 1 and the outgoing arcs are the production variables for period 1 zv_{ij1k}, $k = 1, 2, 3$ and the initial inventory variables w_{ik}, $k = 1, 2, 3$. Some variables have the same source and sink, e.g. zv_{ij11} and w_{i1}. There is, however, an important distinction as the set up forcing (2.9) and capacity constraints (2.10) are only on the production variables and not on the initial inventory variables. For nodes two and three we have the flow conservation constraints (2.8) that are represented by the Figures 2.3 e 2.4: the sum of the flows going into a node must be equal to the sum of the flows going out. The constraint for the final node imposes that the sum of the incoming flows, $w_{i3} + \sum_{k=1}^{3} zv_{ijkm}$, must be equal to one. This constraint is not represented in the model (2.6)-(2.12) as it is redundant.

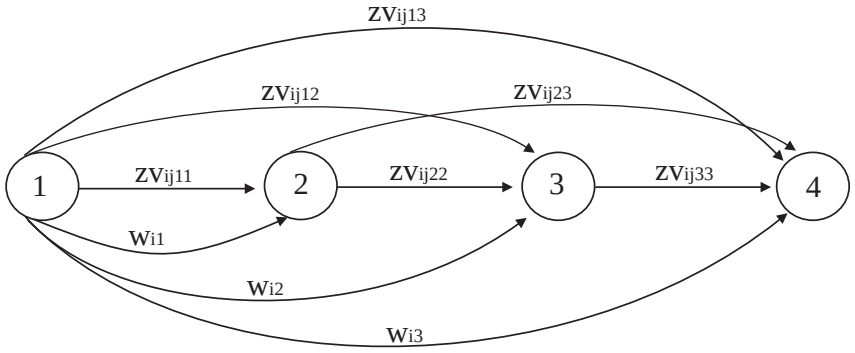


Figure 2.1: Network representation for one item, one machine and 3 periods.

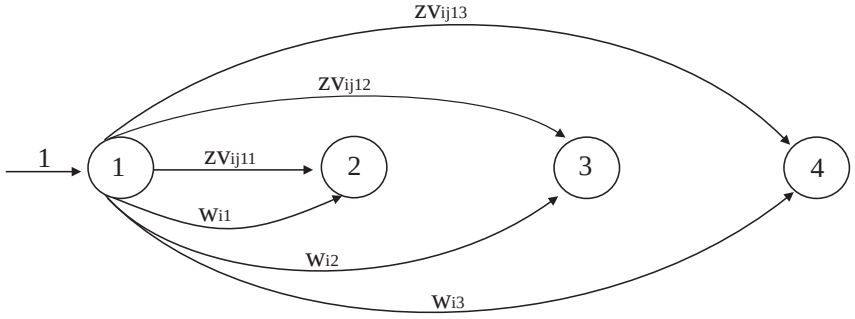


Figure 2.2: Network representation for period 1 constraint: $1 = \sum_{k=1}^{3}(w_{ik} + zv_{i,j,1,k})$

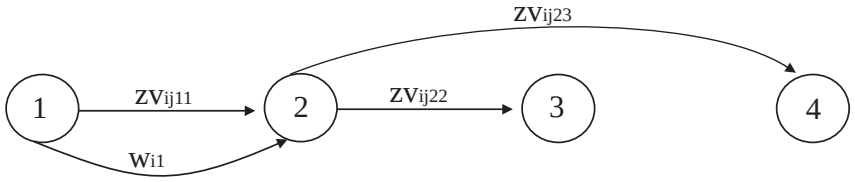


Figure 2.3: Network representation for period 2 constraint: $w_{i1} + zv_{ij11} = \sum_{k=2}^{3} zv_{ij2k}$.

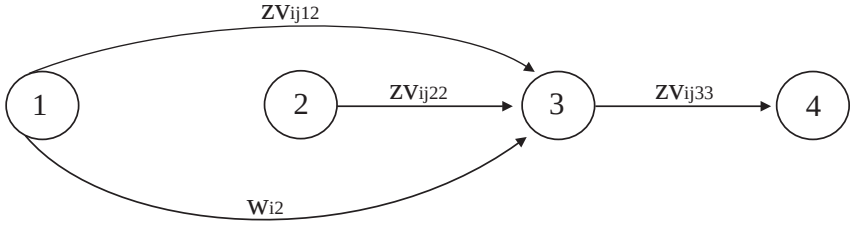


Figure 2.4: Network representation for period 3 constraint: $w_{i2} + \sum_{k=1}^{2} zv_{ijk2} = zv_{ij33}$.

The correspondences between the variables of reformulation (2.6)-(2.12) with the classical formulation (2.1)-(2.5) are as follows:

$$x_{ijt} = \sum_{k=t}^{m} sd_{itk} zv_{ijtk} \qquad \forall i \in I, j \in J, t \in T$$

$$s_{i0} = \sum_{t=1}^{m} w_{it} sd_{i1t} \qquad \forall i \in I$$

We observe again that this model is an adaptation of the shortest path reformulation that was originally proposed by Eppen and Martin (1987) for the case without capacity constraints. Due to the capacity constraints, the variables w_{it} and zv_{ijtk} can be fractional. The interpretation of a fractional value, for instance 0.3 for the variable zv_{ijtk} is as follows:

produce in period t, on machine j, 30% of the total demand from period t until period k.

2.3 Solution method

In this section we present a Lagrangian heuristic that considers the model (2.6)-(2.12) in which the flow constraints (2.7) and (2.8) are dualized in the objective function. The problem decomposes into independent sub-problems, per period and per machine and contains the capacity and setup constraints and the integrality requirements. The motivation for studying this approach comes from the results obtained in Jans and Degraeve (2004a) for the single machine problem. These results showed that the lower bounds that are found using decomposition per period (dualizing the flow constraints) applied to the reformulation are better than those obtained using decomposition per item (dualizing the capacity constraints) applied to the classical formulation.

For the single machine problem, in addition to Jans and Degraeve (2004a), the work of Diaby et al. (1992) and Pimentel et al. (2010) tested the decomposition per period using the classical formulation. In Chen and Thizy (1990) they prove that, for the single machine problem and without setup times, the lower bound obtained through decomposition per period for their reformulation is better than that obtained by decomposition per item in the classical formulation. In Süral et al. (2009), decomposition per period is applied to a reformulation of the problem with a single machine and without setup costs. The authors come to the conclusion that, for the reformulation, decomposition by period is better than decomposition per item. It should be noted that we have not found any paper in the literature that considers such a decomposition per period and machines for the unrelated parallel machines case.

In general, the solution method can be summarized in the following steps:

General Steps for the Lagrangian Heuristic

- Step 0: Attribute initial values to the Lagrange multipliers. Consider the current lower bound to be $-\infty$ and the current upper bound to be $+\infty$;

- Step 1: **Lagrangian Relaxation (Lower Bound)**
  Apply Lagrangian relaxation to the flow constraints (2.7) and (2.8), obtaining independent sub-problems per period and machine. For each period and machine, solve the sub-problem, using the branch-and-bound method proposed by Jans and Degraeve (2004a). Update the lower bound if it is better than the current one;

- Step 2: **Feasibility Heuristic (Upper Bound)**
  If the Lower Bound has been improved, apply the feasibility heuristic containing a

backward, forward and transfer stage to try to find a feasible solution. Update the upper bound if it is better than the current one;

- Step 3: Update the Lagrange multipliers using the subgradient optimization method. Go to Step 1.

2.3.1 Lagrangian relaxation (lower bound)

Throughout this section, we discuss the Lagrangian relaxation applied to the compact formulation (2.6)-(2.12) as developed for the lot sizing problem on parallel machines. In this Lagrangian relaxation, the constraints (2.7) and (2.8) are dualized in the objective function (2.6) with Lagrangian multipliers p_{it}. We define p as the vector of all p_{it} values. Thus, the objective function of the Lagrangian relaxation ($v_{LR}(p)$) is as follows:

$$v_{LR}(p) = Min \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m} sc_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}\sum_{k=t}^{m} cv_{ijtk}zv_{ijtk} - \sum_{i=1}^{n} p_{i1}(\sum_{k=1}^{m} w_{ik} + \sum_{j=1}^{r}\sum_{k=1}^{m} zv_{ij1k} - 1) - \sum_{i=1}^{n}\sum_{t=2}^{m} p_{it}(\sum_{j=1}^{r}\sum_{k=t}^{m} zv_{ijtk} - w_{i,t-1} - \sum_{j=1}^{r}\sum_{k=1}^{t-1} zv_{ijk,t-1}) \quad (2.13)$$

After reorganizing the terms of the objective function, the Lagrangian problem becomes:

$$v_{LR}(p) = Min \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m} sc_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{t=1}^{m-1}(ci_{it} - p_{i,1} + p_{i,t+1})w_{it} + \sum_{i=1}^{n}(ci_{im} - p_{i,1})w_{im} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}\sum_{k=t}^{m-1}(cv_{ijtk} - p_{it} + p_{i,k+1})zv_{ijtk} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}(cv_{ijtm} - p_{it})zv_{ijtm} + \sum_{i=1}^{n} p_{i1} \quad (2.14)$$

Subject to :

$$\sum_{k=t}^{m} zv_{ijtk} \leq y_{ijt} \qquad \forall i \in I, j \in J, t \in T \quad (2.15)$$

$$\sum_{i=1}^{n} st_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{ijt}sd_{itk}zv_{ijtk} \leq Cap_{jt} \qquad \forall j \in J, t \in T \quad (2.16)$$

$$y_{ijt} \in \{0,1\}, \quad 0 \leq w_{it} \leq 1 \qquad \forall i \in I, j \in J, t \in T \quad (2.17)$$

$$zv_{ijtk} \geq 0 \qquad \forall i \in I, j \in J, t \in T \ \ \forall k \in T, k \geq t \quad (2.18)$$

The Lagrangian problem can be decomposed into independent subproblems for each period $t \in T$ and each machine $j \in J$:

$$z_{LR_{tj}}(p) = Min \sum_{i=1}^{n} sc_{ijt} y_{ijt} + \sum_{i=1}^{n} \sum_{k=t}^{m-1} (cv_{ijtk} - p_{it} + p_{i,k+1}) zv_{ijtk} + \sum_{i=1}^{n} (cv_{ijtm} - p_{it}) zv_{ijtm} \quad (2.19)$$

Subject to :

$$\sum_{k=t}^{m} zv_{ijtk} \leq y_{ijt} \qquad \forall i \in I \qquad (2.20)$$

$$\sum_{i=1}^{n} st_{ijt} y_{ijt} + \sum_{i=1}^{n} \sum_{k=t}^{m} vt_{ijt} sd_{itk} zv_{ijtk} \leq Cap_{jt} \qquad (2.21)$$

$$y_{ijt} \in \{0,1\}, \quad zv_{ijtk} \geq 0 \qquad \forall i \in I, \ \forall k \in T, k \geq t \qquad (2.22)$$

The w_{it} variables are present in the Lagrange objective function (2.14), but not in the objective function of the separate subproblems (2.19) because they do not appear in the subproblem constraints. To minimize the overall Lagrange objective function (2.14), the value of each w_{it} variable is determined according to the following decision rule:

$$w_{it} = 1 \text{ if } (ci_{it} - p_{i,1} + p_{i,t+1}) < 0, \ w_{it} = 0 \text{ otherwise} \quad \forall i \in I, t \in T \setminus \{m\} \quad (2.23)$$

$$w_{im} = 1 \text{ if } (ci_{im} - p_{i,1}) < 0, \ w_{im} = 0 \text{ otherwise} \quad \forall i \in I \quad (2.24)$$

These subproblems can be solved by the branch-and-bound method proposed by Jans and Degraeve (2004a). The main idea of this method is that the linear relaxation of the subproblems (2.19)-(2.22) is a Linear Multiple Choice Knapsack Problem (LMCKP). The LMCKP is an extension to the binary knapsack problem, in which the items are divided into some disjoint classes and the problem is to choose exactly one item from each class so that the profit is maximized without exceeding the capacity of the knapsack. Therefore, on each node of the branch-and-bound method proposed by Jans and Degraeve (2004a) solve a LMCKP problem (using a greedy algorithm proposed by Sinha and Zoltners (1979)). The LMCKP structure remains after branching.

The Lagrange multipliers p_{it} are updated by the subgradient optimization method (Camerini et al., 1975). Let p_{it}^{κ} be the Lagrange multipliers at iteration κ and let $(y_{ijt}^{\kappa}, zv_{itk}^{\kappa}, w_{it}^{\kappa})$ the optimal solution for the Lagrangian problem at iteration κ. The Lagrange Multipliers p_{it}^{κ} are updated according to the subgradient scheme (2.25) and (2.26). In the calculation of t_k (2.27) ub is the best known upper bound for the original and α is initially set to one and is decreased whenever the Lagrange solution ($v_{LR}(p)$) has failed to improve in a specified number of steps (the theoretical ideas of the subgradient optimization method are presented in the Section A.1 of the Appendix A).

$$p_{i1}^{\kappa+1} = max\{0, p_{i1}^{\kappa} - t_k(\sum_{k=1}^{m} w_{ik}^{\kappa} + \sum_{j=1}^{r}\sum_{k=1}^{m} zv_{i,j,1,k}^{\kappa} - 1)\} \qquad \forall i \in I \tag{2.25}$$

$$p_{it}^{\kappa+1} = max\{0, p_{it}^{\kappa} - t_k(\sum_{j=1}^{r}\sum_{k=t}^{m} zv_{ijtk}^{\kappa} - w_{i,t-1}^{\kappa} - \sum_{j=1}^{r}\sum_{k=1}^{t-1} zv_{ijk,t-1}^{\kappa})\} \ \forall i, t/\{1\} \tag{2.26}$$

$$t_k = \frac{\alpha(ub - v_{RL}(p_{it}^{\kappa}))}{\sum_{i=1}^{n}(\sum_{k=1}^{m} w_{ik}^{\kappa} + \sum_{j=1}^{r}\sum_{k=1}^{m} zv_{i,j,1,k}^{\kappa} - 1)^2 + \sum_{i=1}^{n}\sum_{t=2}^{m}(\sum_{j=1}^{r}\sum_{k=t}^{m} zv_{ijtk}^{\kappa} - w_{i,t-1}^{\kappa} - \sum_{j=1}^{r}\sum_{k=1}^{t-1} zv_{ijk,t-1}^{\kappa})^2} \tag{2.27}$$

The value of the objective function for the solution of the relaxed problem ($v_{LR}(p)$) produces a lower bound for the original problem and the Lagrangian Dual problem gives the maximum lower bound $v_{DL} = max_p v_{LR}(p)$.

To start the subgradient optimization method, fix the dual variables as zero ($p_{it}^0 = 0$) and make the size of the step in the direction of the subgradient proportional to a parameter α that must decrease at each iteration. The rule used to generate a decreasing sequence for α is characterized by parameters (α_0, r_0, d). The values of α are obtained by making $\alpha = \alpha_0$ and it is progressively decreased by multiplying it by a parameter d, if the Lagrangian solution is not improved in the last $r = r_0$ iterations. After extensive computational tests, with a view to defining the best combination of these parameters, the defined values are $\alpha_0 = 1, r_0 = 50, d = 0.6$ and 2500 iterations of the method are made.

2.3.2 Feasibility heuristic (upper bound)

To obtain a feasible solution (upper bound) a feasibility heuristic is used. The initial solution is given by solving the Lagrangian relaxation applied to the flow constraints (2.7) and (2.8). The solution of the subproblems are then grouped and, generally, the resulting solution is not a feasible solution for the problem (2.6)-(2.12), due to the fact that the flow constraints were not taken into consideration. In other words, the solution probably does not satisfy the demand for all items and periods.

If the initial solution does not satisfy the demand constraints, the feasibility heuristic (containing a backward stage, a forward stage and if necessary a transfer stage) is applied over all the periods, making changes to the production plan, producing and, if necessary, removing excess production in its attempt to try to make the solution a feasible one (there is no guarantee that we will always get a feasible solution). Note that feasibility heuristics based on production transfer have been applied to make the capacity constraints feasible

(for instance, Trigeiro et al. (1989) and Toledo and Armentano (2006)). However, in the literature we have not found any feasibility heuristics, based on transfers of production, to make the demand constraints feasible. In general, the feasibility heuristic steps can be described as follows:

Feasibility Heuristic Steps

- Step 2.1: **Backward Stage (free up capacity)**
  For each item, find the total production quantity and check if production is in excess of demand. If it is, starting with the last period moving to the first, remove excess production from those periods where the item is produced;

- Step 2.2: **Forward Stage**
  Starting with the first period and moving to the last, check if demand is met and, if it is not, try to make the solution feasible in the following order:

  - Produce the quantity still needed with machines already set up (no new setups);
  - Produce the remaining quantity adding new machines setup;

- Step 2.2.1: **Transfer Stage**
  If, after all the above attempts, demand is still not met for a given item i', try this last one, which checks among previously-made-feasible items for one which has excess production and removes this excess production, thus freeing up capacity. Try to satisfy the demand for the given item i', by using this capacity.

- Step 2.3: **Backward Stage (remove possible remaining excess)**
  After applying the Forward and Transfer Stages, some production excess can arise again. More specifically, the first application of the Backward Stage guarantees that the total production (for all periods) is less or equal to the total demand (for all periods). However, it does not guarantee that the demand of each individual period is met. So, the subsequent Forward and Transfer Stages can produce some items to meet this demand and then we have excess again and the Backward Stage is applied to remove this excess.

Following, each stage is described in detail.

• Backward Stage

This first stage to make a solution feasible consists of freeing up capacity if we have excess production, and to do this, the production quantities for each item are analyzed (and removed if necessary) from the last to the first period. Initially we calculate:

$$\Delta(i) := \sum_{j=1}^{r}\sum_{t=1}^{m} x_{ijt} - \sum_{t=1}^{m} d_{it} \;\; i = 1, 2, ..., n$$

If $\Delta(i) > 0$ for some $i = 1, 2, ..., n$ the production excess of this item must be eliminated. To do this, starting with the last period m and following the sequence $t = m, m-1, ..., 1$, for a period t where there is production of item i, look for a machine j with the greatest production cost and then calculate:

$$Min\{x_{ijt}, \Delta(i)\}$$

If $Min\{x_{ijt}, \Delta(i)\} = x_{ijt}$ remove the total amount produced and consequently the setup for this machine and period. Thus, the remaining capacity is updated and then move on to the previous period to remove any remaining excess production. Denoting by Cap'_{jt} the remaining capacity, the update is given by:

$$Cap'_{jt} := Cap'_{jt} + (x_{ijt} \times vt_{ijt}) + st_{ijt}$$

The values of $\Delta(i)$ and the variables are also updated by:

$$\Delta(i) := \Delta(i) - x_{ijt}; \;\; x_{ijt} := 0; \;\; y_{ijt} := 0$$

However, if $Min\{x_{ijt}, \Delta(i)\} = \Delta(i)$, only remove part of the total amount produced, i.e., enough to remove the excess production and update the values by:

$$Cap'_{jt} := Cap'_{jt} + (\Delta(i) \times vt_{ijt}); \;\; x_{ijt} := x_{ijt} - \Delta(i); \;\; \Delta(i) := 0$$

In Figure 2.5 we provide an example with two items, two machines and four periods. In part (a) we see a possible solution of the subproblems in which there is excess of production for item 1 in the third and fourth period. The sum of these excesses will give the value of $\Delta(1)$. Note that in period four there is production of the item 1 on both machines. We assume that the production cost of the item 1 on machine 1 is higher than machine 2. Furthermore, $Min\{x_{114}, \Delta(1)\} = \Delta(1)$, in other words, we will remove all excess of the item 1 using the period four and machine 1. In part (b) we see the production and machines levels after applying the backward stage (remove the production excess). Note that we are not meeting the demand of the item one in period four anymore.

• **Forward Stage**

When the process of eliminating the excess production is over, start to check if the demand is met and try to make feasible possible pairs of item and period (i, t) that do not meet the demand. For this, the analysis will be done from period 1 through to the last period $(t = 1, 2, ..., m)$.

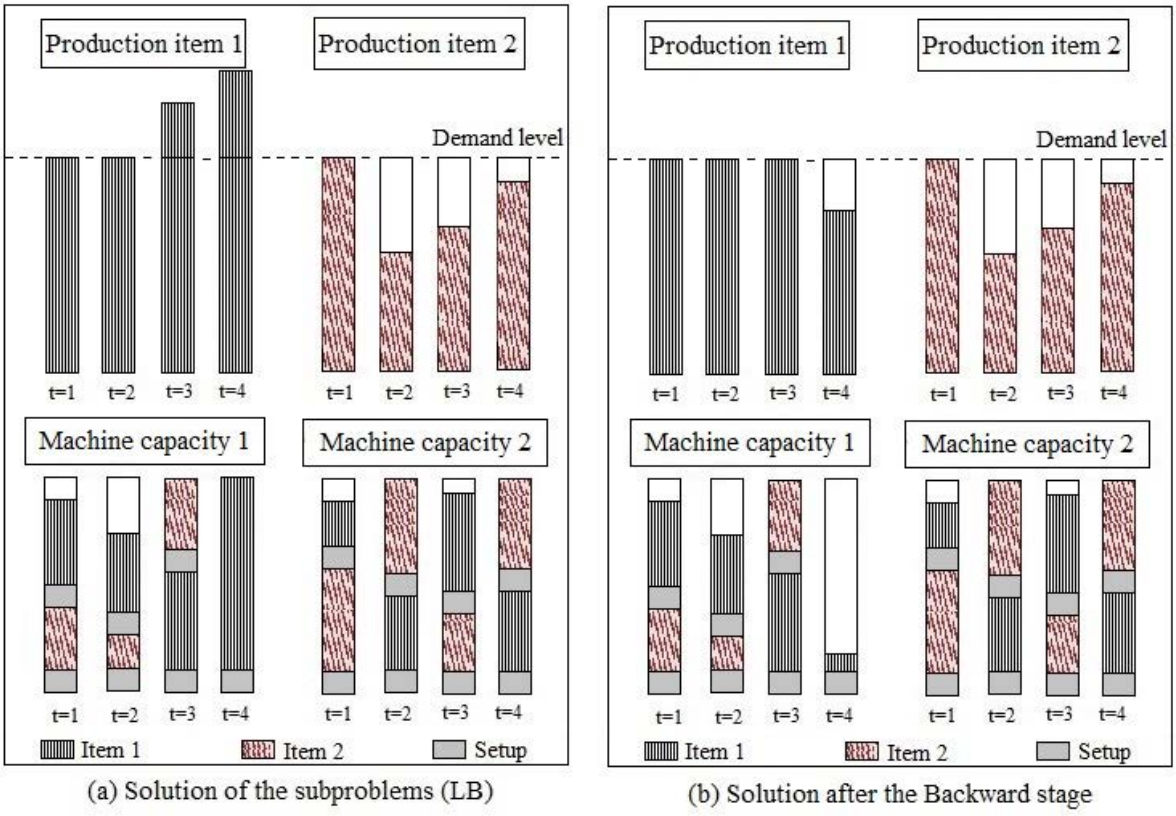


Figure 2.5: Example of the backward stage.

To check if demand is met, an vector $\Omega(i)$, $i = 1, 2, ..., n$ is needed that counts the total production for each item up to the given period. Initially $\Omega(i) = 0$ for $i = 1, 2, ..., n$. Starting from period $t = 1$ (fix the period), go through all the items calculating $\Omega(i) := \Omega(i) + \sum_{j=1}^{r} x_{ijt}$ and do the following analysis.

If $\Omega(i) \geq d_{it}$ the demand for (i, t) is met. Simply remove this specific demand from the total production, in other words, $\Omega(i) := \Omega(i) - d_{it}$ and move on to analyze the next item. Observe that in this case we have $\Omega(i) \geq 0$ and at this point $\Omega(i)$ counts the total inventory.

If $\Omega(i) < d_{it}$, demand is not met for this pair (i, t), then the feasibility process needs to be started. The quantity of items that need to be produced is easily calculated:

$$\Phi = d_{it} - \Omega(i)$$

Once this amount has been calculated, start at the current period t and go back period by period to the first period ($\tau = t, t-1, ...1$) checking first if the production plans that are 1 (with the machine already set up for this item ($y_{ij\tau} = 1$)) have sufficient capacity to produce the quantity that is still needed to satisfy demand.

In other words for $\tau = t, t-1, ...1$ and $j = 1, ..., r$ do:

If $y_{ij\tau} = 1$, that is, if the machine is already set up, analyze the whole amount of items to be produced on this machine in this period:

$$Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\}$$

Then, if $Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\} = \Phi$ the machine can produce the required amount to satisfy demand for this pair (i,t). Therefore, the values are updated by:

$$Cap'_{j\tau} := Cap'_{j\tau} - \Phi \times vt_{ij\tau};\ \ x_{ij\tau} := x_{ij\tau} + \Phi;\ \ \Phi := 0;\ \ \Omega(i) := 0$$

However, if $Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\} = \frac{Cap'_{j\tau}}{vt_{ij\tau}}$, there is still a quantity which needs to be produced, so we update the values and move to the next machine or the previous period and repeat the same analysis.

To update the values in this case do:

$$Cap'_{j\tau} := 0;\ \ x_{ij\tau} := x_{ij\tau} + \frac{Cap'_{j\tau}}{vt_{ij\tau}};\ \ \Phi := \Phi - \frac{Cap'_{j\tau}}{vt_{ij\tau}}$$

Figure 2.6 shows the small example again. In part (b) we see again the production and machines levels after applying the backward stage. For the item 1 we have $\Omega(1) < d_{14}$ and then, as $y_{114} = 1$, we use part of the machine capacity 1 in the fourth period to meet the demand of the item 1 in the fourth period. For the item 2 we have $\Omega(2) < d_{22}$ and we use the hold machine capacity 1 in the second period ($y_{212} = 1$) and in the first period ($y_{211} = 1$) to meet the demand of this period. After update the values we see that we still have $\Omega(2) < d_{23}$ and we use the hold machine capacity 2 in the third period ($y_{223} = 1$) and in the first period ($y_{221} = 1$) to meet this demand. Finally, we have $\Omega(2) < d_{24}$. Note that we can not meet the demand of the item 2 in the fourth period using only the machines already set up. In part (c) we see the updated production and machine levels.

At the end of this first attempt to make it feasible, if there is not sufficient capacity with the pre-established production plans, open up the possibility of adding the setup of another machine such that it can produce all of the remaining quantity required to satisfy demand.

Thus check if there is a machine j such that

$$Cap'_{j\tau} \geq \Phi \times vt_{ij\tau} + st_{ij\tau}$$

If there is more than one machine that satisfies the inequality above, choose the machine with the lowest cost and adjust the production accordingly:

$$x_{ij\tau} := x_{ij\tau} + \Phi;\ \ y_{ij\tau} := 1;\ \ Cap'_{j\tau} := Cap'_{j\tau} - (\Phi \times vt_{ij\tau}) - st_{ij\tau};\ \ \Omega(i) := 0$$

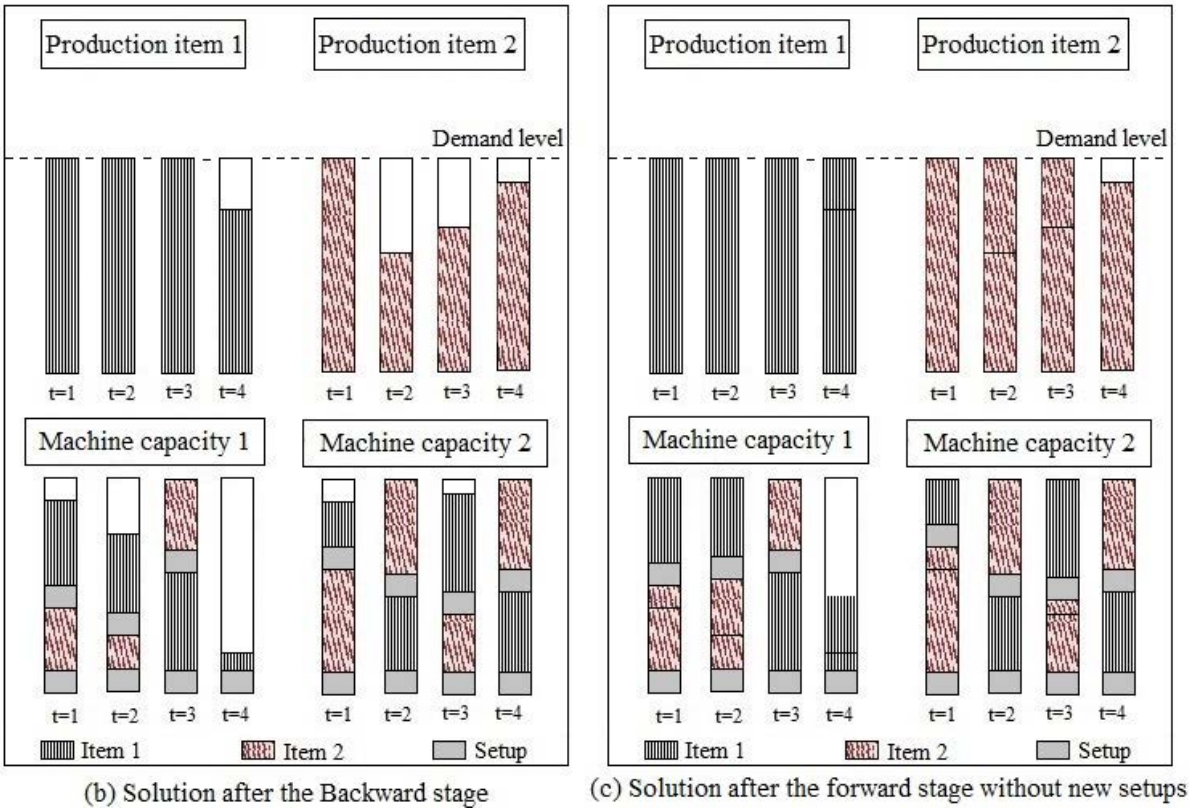


Figure 2.6: Solution after the forward stage without new setups.

However, if there is still no machine with enough capacity, open up the possibility of setting up more machines. At this stage, the criteria used to determine the order of which machine to set up is the relation between the machine's capacity and the total cost of production of the required quantity, in other words:

$$\Re_j = \frac{Cap'_{j\tau}}{\Phi \times vc_{ij\tau} + sc_{ij\tau}}$$

Thus, the first machine that will be set up in the given period is that which has the largest value $\Re_j$ prioritizing the machine with the largest capacity per unit of production cost.

If all the items are made feasible in the current period, move on to the next period and repeat the procedure. If not, i.e., one of the items is still not feasible, there is the Transfer Stage.

Figure 2.7 illustrates the example after the forward stage. We have included a new setup of the item 2 on machine 1 in the fourth period to produce the required amount to meet the demand of the item two in period four. Finally, in part (d) we see the production and capacity level of a feasible solution.

- **Transfer stage**

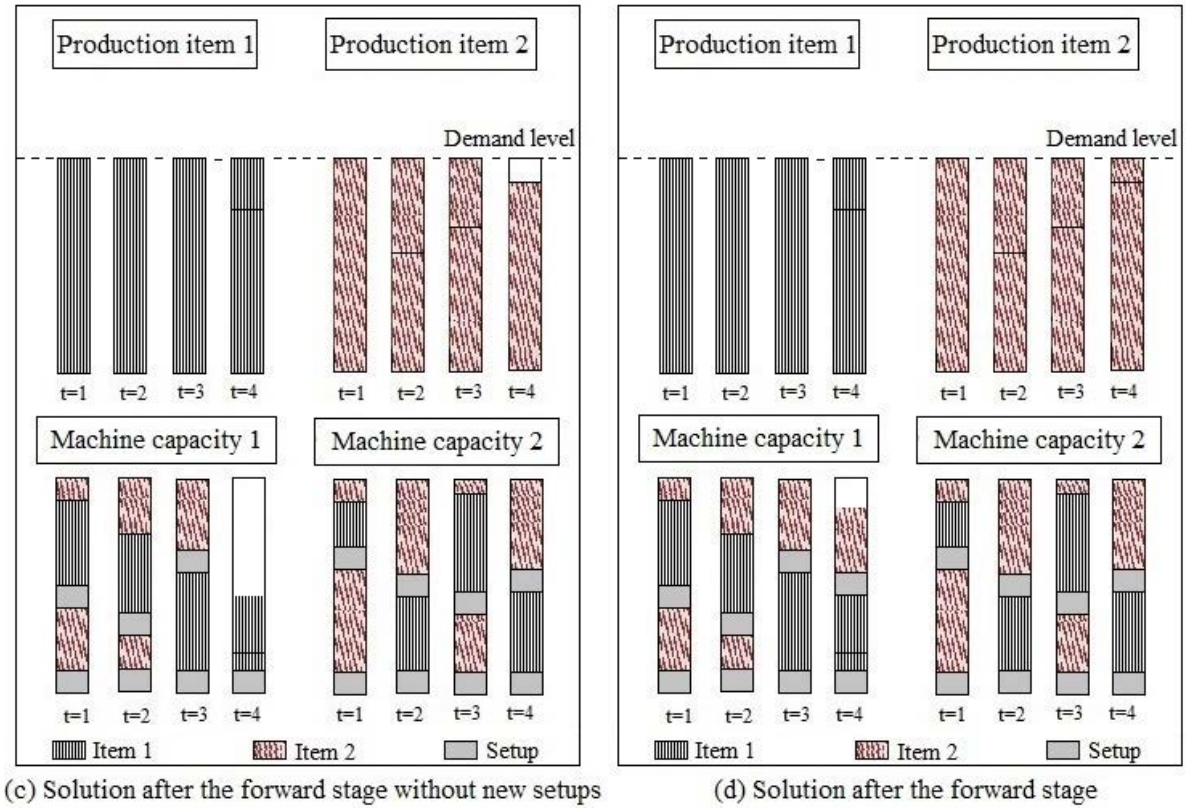


Figure 2.7: Example of the forward stage.

If all the attempts outlined above have been made and it is still not feasible, then apply this last attempt to make it feasible. Where, after all the previous attempts, it is still not possible to meet the demand for a given item i in a given period t, look among previous items for one that has inventory, i.e., the value of Ω is positive. If it is found, make the necessary adjustments, i.e., remove the excess production of the found item (freeing up capacity) and try to produce the required amount of item (i).

Thus, searching in the following order $\tau = t, t-1, ...$ and $i' = i-1, i-2, ..., 1$ we analyze all pairs of items and periods.

If $\Omega(i') > 0$ and $y_{i'j\tau} = 1$, in other words, an item with a quantity in inventory $(\Omega(i') > 0)$ and a machine and period already set up for this item was found, we have two possibilities.

I) If $\Omega(i') \geq x_{i'j\tau}$ then, as well as freeing up capacity, a setup of the machine can be removed i.e.:

$$Cap'_{j\tau} := Cap'_{j\tau} + x_{i'j\tau} \times vt_{i'j\tau} + st_{i'j\tau};\ \ y_{i'j\tau} := 0;\ \ \Omega(i') := \Omega(i') - x_{i'j\tau};\ \ x_{i'j\tau} := 0$$

II) However, if $\Omega(i') < x_{i'j\tau}$, then just free up the capacity:

$$Cap'_{j\tau} := Cap'_{j\tau} + \Omega(i') \times vt_{i'j\tau};\ \ x_{i'j\tau} := x_{i'j\tau} - \Omega(i');\ \ \Omega(i') := 0$$

With the capacity freed, calculate the total quantity of items (i) that can be produced and check again to see if it is possible to produce the required amount to meet demand. It is necessary to consider two possibilites:

If the machine is already set up for the item i, then calculate:

$$Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\}$$

However, if there is not set up for the item i calculate:

$$Min\{\Phi, \frac{(Cap'_{j\tau} - st_{ij\tau})}{vt_{ij\tau}}\}$$

After calculated the minimum considering one of these possibilities if this minimum is Φ, a quantity sufficient to satisfy the demand for this pair (i, τ) can be produced and so adjust the production amount and set up accordingly.

However, if the minimum is $\frac{Cap'_{j\tau}}{vt_{ij\tau}}$ or $\frac{(Cap'_{j\tau} - st_{ij\tau})}{vt_{ij\tau}}$, there is still a quantity to be produced and so remove the quantity that can be produced and move on to the previous item (already made feasible) with a quantity in inventory and check the same conditions.

Once these feasibility stages have been performed for all items, move on to the next period of the forward stage and start the analysis again, for each item from first to last, trying to make the solution feasible. Repeat this procedure until the last period of the forward stage.

Figure 2.8 illustrates a situation in which the forward stage does not make the solution feasible and thus shows the necessity of the transfer stage. In part (a) we see another possible solution for the subproblems. Note that since $\Delta(1) = 0$ and $\Delta(2) < 0$ we do not have excess of production then we do not need to remove any item in the backward stage. Note also that we are producing a large quantity of the item 1 in the first period so that it has utilized all capacity of the machines 1 and 2 in the first period. Therefore, with the forward stage it is not possible to produce the remaining amount to meet the demand of the item 2 in the first period. In this case we move to the transfer stage. As we have $\Omega(1) > 0$ and we are producing the item 1 on machine 1 in the first period, it is possible to free up the capacity of this machine in this period (reducing the production of the item 1) and produce the required amount to meet the demand of the item 2 in the first period. We also observe that $\Omega(1) > x_{111}$ then we can remove the setup of the item 1 on machine 1. In part (b) we see the production and machine levels after applying the transfer stage in the first period.

Finally, at the end of the forward stage, re-apply the backward stage to remove any possible excesses that still remain and calculate the value of the solution, creating an upper bound (feasible solution) if the solution becomes feasible, i.e., if demand is met.

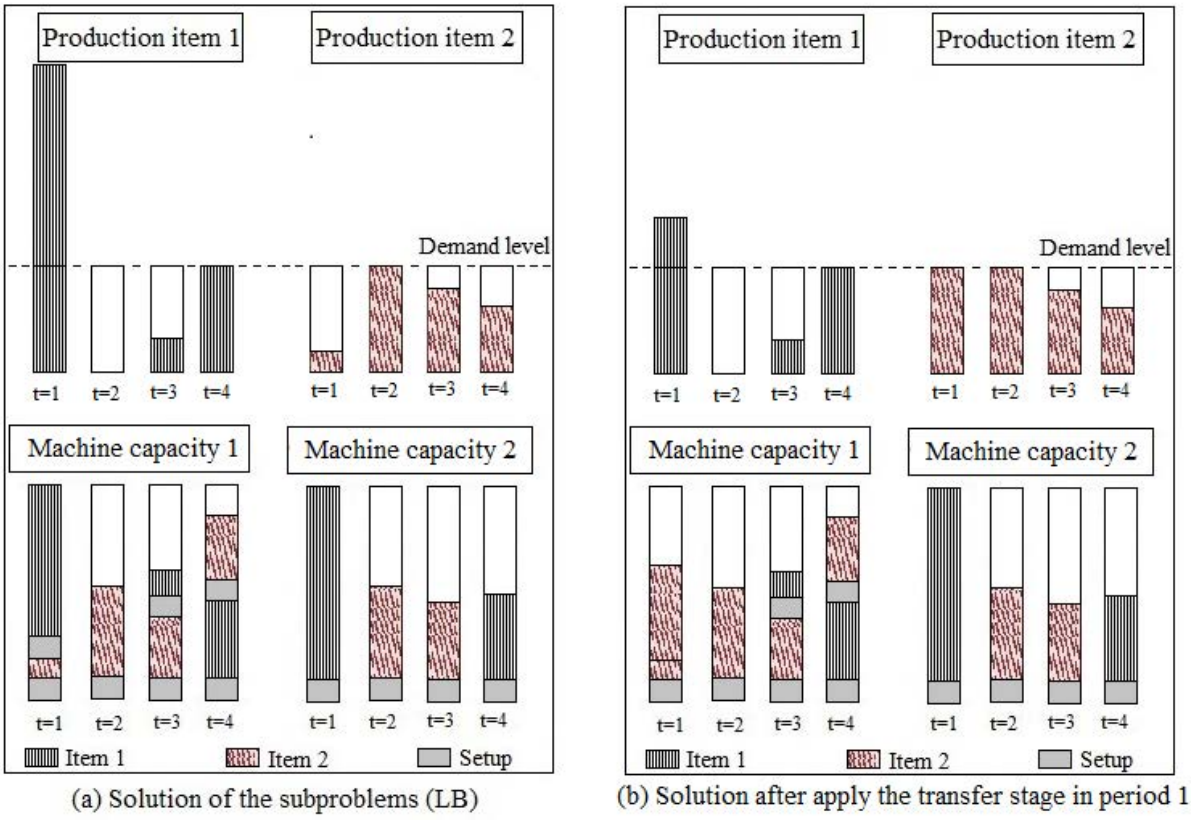


Figure 2.8: Example of the transfer stage.

2.4 Computational results

The heuristic was programmed in Fortran and the tests were done on a personal computer Intel Core-I5, 2.27GHz with 6GB of RAM and the Windows operating system. The computational tests involve two experiments with instances given in Toledo and Armentano (2006). In the first experiment, the proposed heuristic called FA is tested for small instances and the quality of the solutions is compared with the heuristic proposed in Toledo and Armentano (2006) (TA) which consists of a Lagrangian heuristic based on Trigeiro et al. (1989) in which the capacity constraints are relaxed. In the second experiment, the heuristic is tested for large instances and, in addition to comparing the results with those obtained in Toledo and Armentano (2006), it is also compared (for instances with 18 periods) with results from the CPLEX 12.2 software package, applied to the formulation (2.6)-(2.12).

2.4.1 Results for experiment 1

The heuristic was tested for a set of 80 small instances proposed in Toledo and Armentano (2006) with $n = 8$ items, $r = 2$ machines and $m = 4$ periods. The heuristic

was evaluated using the quality of its solution and the quantity of feasible solutions found and analyzed according to three factors: setup cost, setup time and capacity. The setup costs and times can be high (H) and low (L) and the capacity can be normal (N) or tight (T). For each combination of these three factors, ten instances were generated and, for example, the configuration NLH means normal capacity, low setup cost and high setup time.

The parameters were generated from intervals $[a, b]$ with uniform distribution, $U[a, b]$:

- production cost (vc_{ij}) U[1.5,2.5];
- setup cost (sc_{ij}) U[5.0,95.0];
- inventory cost (hc_i) U[0.2,0.4];
- production time (vt_{ij}) U[1.0,5.0];
- setup time (st_{ij}) U[10.0,50.0];
- demand (d_{it}) U[0,180];

To generate instances with high setup costs we multiply the setup costs by 10. In the same way, to generate instances with high setup times we multiply the setup times by 1.5.

The capacity was also generated in the same way as Toledo and Armentano (2006), that is, using as its basis the capacity needed to produce the items according to a lot-for-lot policy:

$$Cap = \frac{\sum_{t=1}^{m}\sum_{j=1}^{r}\sum_{i=1}^{n}(\frac{d_{it}}{r}vt_{ij} + st_{ij})}{rm}$$

Afterwards, an adjustment is made to reduce the capacity in order to generate instances which use about 80% of the capacity ($Cap = (1.18 - 0.07r)Cap$). The tight capacity is obtained by multiplying this capacity by 0.9.

We observe that in the tables in which we compare the gaps, it is calculated using the formula $gap = \frac{(UB-LB)*100}{LB}$, in which UB and LB denote the upper and the lower bounds found by the solution approach. Moreover, the columns $FEA(\%)$ indicate the percentage of feasible solutions found by each method.

Table 2.1 shows the heuristics's behavior for the set of small instances generated. Note that the proposed heuristic obtained better gaps in all the configurations and that the difference significantly increases in problems with high setup costs, since, if the setup cost is high the tendency is for the initial solution to have few machine setups, making

the feasibility phase for the heuristic in Toledo and Armentano (2006) more difficult, because this dualizes the capacity constraints. Note that both heuristics get better results for problems with low setup costs. With regard to the percentage of feasible solutions found (FEA), once again the question of capacity relaxation (Toledo and Armentano 2006) is relevant, so that for the configuration THH, the heuristic proposed by Toledo and Armentano (2006) only finds 70% of feasible solutions, whilst this problem does not happen for the heuristic proposed here where the demand constraints are dualized, meaning feasible solutions are found for all the instances. In relation to computational times, both heuristics take less than one second for all the instances. However it should be noted that there is a difference between the machines used. The results from Toledo and Armentano (2006) were obtained with an AMD Athlon XP2600 with 2.08GHz and 1GB of RAM.

	TA		FA	
capacity/setup	FEA(%)	GAP	FEA (%)	GAP
NLL	100	1.5	100	0.7
TLL	93	3.2	100	1.4
NHL	100	15.8	100	4.3
THL	99	32.6	100	4.8
NLH	100	2.1	100	0.5
TLH	73	5.4	100	1.1
NHH	100	20.6	100	5.7
THH	70	41.7	100	6.0

Table 2.1: Relative gap for small instances with two parallel machines

2.4.2 Results for experiment 2

In this experiment, the heuristic was tested for a total of 2880 instances, such that 10 instances were generated for each configuration of number of items ($n = 6, 12, 25, 50$), machines ($r = 2, 4, 6$) and periods ($m = 6, 12, 18$) and in which the parameters were generated in the same way as in Experiment 1. Table 2.2 shows the performance of the heuristic for all sizes according to three factors: capacity, setup cost and setup time. The parts of the table concerning the gaps and time represent the average of all 360 instances generated for each configuration. The high percentage of feasible solutions found by the two heuristics show that both methods are efficient for these classes of problems, however, once again, the proposed heuristic here obtained better quality gaps and found more feasible solutions. Note that the biggest gaps are associated with high setup cost problems, in the same way as seen in Table 2.1. On the other hand, the setup time does

not so much affect the quality of the solutions.

	TA			FA		
capacty/setup	FEA(%)	GAP	Mean time	FEA(%)	GAP	Mean time
NLL	99	2.0	3.9	100	1.4	3.3
TLL	98	3.4	8.9	100	2.3	4.1
NHL	99	11.7	5.3	100	8.9	4.9
THL	98	18.5	7.8	100	12.3	6.5
NLH	98	2.5	4.9	100	1.5	3.5
TLH	95	4.2	14.4	99	2.6	4.7
NHH	98	13.8	5.9	100	9.9	5,2
THH	96	23.5	8.4	99	13.2	6.6

Table 2.2: Heuristic performance for larger instances

Tables 2.3 and 2.4 illustrate the effect caused by the number of items and machines for instances with 12 periods, normal capacity, low setup times and low setup costs (Table 2.3 - class NLL) or high setup costs (Table 2.4 - class NHL). For other numbers of periods (6 and 18) the conclusions were similar to those shown in Tables 2.3 and 2.4 (see Table 2.5). The tables show that the gaps become significantly smaller as the number of items increases, so that the largest gaps found by the two heuristics were for instances with 6 items. In the case of high setup costs, 6 items and 6 machines, the gaps found by the heuristic from Toledo and Armentano (2006) and by the proposed heuristic are 52.8 and 28.1, respectively.

Comparing the results of Table 2.3 to those of Table 2.4, one can see that the gaps and computational times are bigger for problems with a high setup cost which shows a significant increase in the difficulty of problems when the setup costs are increased.

Table 2.5 illustrates the results for instances with 6 and 18 periods for the class NLL. Note that the table contains only the results for the proposed method (FA), because Toledo and Armentano (2006) do not present the results for these periods. As stated previously, the results are similar, i.e., although there are differences for the same configurations (number of items and machines) the results show a same pattern, the largest gaps are for instances with 6 items and the gaps become significantly smaller (in a similar reason) as the number of items increases.

For the comparisons with CPLEX 12.2 in the Table 2.6 (2.6)-(2.12) and only the configurations with 18 periods. For instances with 6 and 12 periods the CPLEX proved optimality for an expressive amount of instances. On the other hand, for the 18 period problems, in most cases, the solver did not prove optimality for any of the instances in a time limit of 120s. These results are shown only for one class of problems since the

		TA			FA		
r	n	FEA (%)	GAP	Mean time	FEA(%)	GAP	Mean time
2	6	99	2.2	0.3	100	1.7	0.5
	12	100	1.0	0.6	100	0.8	1.3
	25	100	0.3	1.5	100	0.5	3.0
	50	100	0.1	2.6	100	0.2	4.9
4	6	100	5.8	0.9	100	3.0	1.0
	12	100	1.8	1.7	100	1.0	1.6
	25	100	0.3	3.0	100	0.2	3.0
	50	100	0.1	6.2	100	0.1	6.3
6	6	90	10.4	1.6	100	6.3	1.0
	12	100	2.4	3.1	100	0.8	1.9
	25	100	0.9	6.9	100	0.2	4.4
	50	100	0.3	14.9	100	0.0	7.7

Table 2.3: Results for instances with 12 periods, normal capacity, low setup cost and low setup time (NLL)

		TA			FA		
r	n	FEA(%)	GAP	Mean time	FEA(%)	GAP	Mean time
2	6	95	18.2	0.5	100	15.4	0.9
	12	100	7.1	0.9	100	5.4	1.7
	25	100	3.4	2.3	100	2.2	5.8
	50	100	1.0	6.1	100	1.5	11.7
4	6	100	31.2	0.9	100	24.3	1.1
	12	100	8.9	2.1	100	11.2	2.0
	25	100	3.1	4.3	100	2.8	4.3
	50	100	1.1	10.3	100	1.1	9.4
6	6	90	52.8	1.7	100	28.1	1.5
	12	100	16.6	3.7	100	13.1	2.7
	25	100	4.9	8.0	100	3.4	5.3
	50	100	1.7	17.4	100	1.1	10.5

Table 2.4: Results for instances with 12 periods, normal capacity, high setup cost and low setup time (NHL)

conclusions are similar for the other classes.

In Table 2.6, the lower bounds found by CPLEX (from linear relaxation-Linear Rel. column) in the root node after applying the cuts (cuts column) and after the branches, in other words, the best lower bound found by the solver (branches column) are compared with those generated by the methods of Toledo and Armentano (2006) (decomposition per

		FA (6 periods)			FA (18 periods)		
r	n	FEA (%)	GAP	Mean time	FEA(%)	GAP	Mean time
2	6	100	2.0	0.3	100	3.1	0.9
	12	100	0.4	0.7	100	2.1	3.4
	25	100	0.4	2.2	100	1.2	7.7
	50	100	0.1	5.4	100	0.2	20.1
4	6	100	1.9	0.3	100	6.2	1.8
	12	100	0.9	0.7	100	1.6	3.7
	25	100	0.2	1.6	100	0.5	7.7
	50	100	0.1	3.7	100	0.2	15.6
6	6	100	4.3	0.3	100	7.7	2.3
	12	100	0.6	0.7	100	1.7	4.6
	25	100	0.2	1.6	100	0.3	9.3
	50	100	0.0	3.3	100	0.1	18.2

Table 2.5: Results for instances with 6 and 18 periods, normal capacity, low setup cost and low setup time (NLL)

item) and the proposal made in this chapter (decomposition per period and per machine).

Note that the proposed FA method found, in all cases, better lower bounds than the TA method. Table 2.6 also shows that in relation to CPLEX, even after branching, the lower bounds found using the FA method are better in most of the 25 and all of the 50 items instances. This means that, if the lower bounds generated by the FA method are used in the root node of CPLEX, in addition to getting better gaps, it may also helps the solver to make various prunes in the solution tree and find feasible solutions.

To compute the gap of CPLEX we use as lower bounds those found in the root node after the cuts have been applied. The gaps found by the solver are better mainly due to the greater efficiency of the solver and the longer computational time, as the solver runs for 120 seconds whilst the average time used by the heuristic is 8.1 seconds for the larger problems.

To finalize the computational tests we solved some new larger instances with 18 periods, 6 machines and up to 200 items for two difficult class: TLH and THH. Table 2.7 shows the results. Note that all lower bounds found by CPLEX and the proposed method are similar and the gaps found by method FA are very good, even for the class THH which is the class that the method FA found the largest gaps on the previous results. Note that the lower bounds found by the CPLEX in the linear relaxation, after applying the cuts and after the branches are very similar, especially for the class TLH. It occurs first because for these instances the lower bounds found by linear relaxation are very good and are not improved after the cuts made by CPLEX and secondly because

			CPLEX (LB)			TA	FA	Gaps	
	r	n	Linear Rel.	cuts	branches	LB	LB	Cplex	FA
TLH	2	6	22031	22084	23239	21925	22173	5.4	7.3
		12	43178	43185	43273	42957	43096	0.2	5.6
		25	89425	89427	89459	89236	89453	0.0	1.3
		50	176720	176735	176745	176519	176748	0.0	0.8
	4	6	20175	20256	20449	20064	20422	1.9	8.2
		12	39897	39957	40046	39933	40085	0.5	3.0
		25	80920	80944	80981	80941	80989	0.0	1.0
		50	162390	162405	162411	162403	162429	0.0	0.5
	6	6	19630	19728	19984	19372	19985	3.1	9.3
		12	38380	38413	38494	38382	38507	0.4	2.6
		25	78798	78819	78853	78808	78868	0.1	0.7
		50	154702	154711	154716	154708	154725	0.0	0.3

Table 2.6: Lower bounds and gaps for 18 periods

for many instances the solver solves the problem at the root node before apply the cuts. Analyzing the computational times, we fixed the CPLEX time limit in 120s again and the general average of the classes TLH and THH are 59.1 and 159.8 seconds, respectively considering the method FA.

			CPLEX (LB)			FA	Gaps	
	r	n	Linear Rel.	cuts	branches	LB	Cplex	FA
TLH	6	75	233452	233452	233458	233442	0.0	0.2
		100	314035	314035	314035	314008	0.0	0.1
		125	392336	392336	392339	392299	0.0	0.1
		150	473797	473797	473797	473799	0.0	0.0
		175	552310	552310	552310	552312	0.0	0.0
		200	629863	629863	629863	629850	0.0	0.0
THH	6	75	330943	330955	331098	331042	0.1	1.1
		100	444464	444473	444681	444527	0.0	0.8
		125	553420	553423	553566	553465	0.0	0.5
		150	668420	668421	668552	668447	0.0	0.5
		175	780575	780577	780645	780602	0.0	0.4
		200	890542	890543	890643	890561	0.0	0.3

Table 2.7: Lower bounds and gaps for large instances with 18 periods

2.5 Conclusion

In this chapter, the lot sizing problem with capacity constraints in parallel machines was studied. A reformulation of the problem was proposed using the strategy of variable redefinition proposed by Eppen and Martin (1987), and a Lagrangian heuristic was developed in which the demand constraints are relaxed, thus the problem is decomposed per period and per machine instead of the classical per item decomposition. A feasibility heuristic is applied with the idea of satisfying the demand constraints. This strategy was compared to the Lagrangian heuristic proposed by Toledo and Armentano (2006) in which the classical decomposition is used and also compared to CPLEX 12.2. The results show the great efficiency of the proposed method in relation to the lower bounds obtained, which are better than those in the literature and competitive with CPLEX. The method also presents competitive upper bounds and, although they are not better those obtained by the CPLEX package, they are better than the results from the literature.

Chapter 3

Hybrid methods for lot sizing on parallel machines

In this chapter we continue to deal with the capacitated lot sizing problem with multiple items, setup time and unrelated parallel machines, and apply Dantzig-Wolfe decomposition to the shortest path reformulation of the problem.[1] Unlike in the traditional approach where the linking constraints are the capacity constraints, we use the flow constraints, i.e. the demand constraints, as linking constraints. The aim of this approach is to obtain high quality lower bounds. We solve the master problem applying two solution methods that combine Lagrangian relaxation and Dantzig-Wolfe decomposition in a hybrid form. Note that the theoretical basis of these solution methods are presented in the Appendix A (hybrid methods). A primal heuristic, based on transfers of production quantities, is used to generate feasible solutions. Computational experiments using data sets from the literature are presented and show that the hybrid methods produce lower bounds of excellent quality and competitive upper bounds, when compared with the bounds produced by other methods from the literature and by a high-performance MIP software.

This chapter has the following contributions. First, we propose a way to obtain lower bounds that are stronger than the ones obtained by the traditional per-item Dantzig-Wolfe decomposition. Second, we extend two hybrid algorithms that combine Lagrangian relaxation and Dantzig-Wolfe decomposition and apply them to obtain the stronger lower bounds for the problem with unrelated parallel machines. Third, we improve the Lagrangian heuristic proposed by Fiorotto and Araujo (2014) to obtain better upper bounds. Finally, computational experiments are performed to show the quality of the upper bounds and lower bounds compared to other methods from the literature. A comparison shows

[1] A compact version of this chapter has been published as a research report (CIRRELT-2014-56) and submitted to an international journal.

that the new hybrid method together with the improved heuristic provides generally better gaps.

3.1 Literature review on Dantzig-Wolfe decomposition applied to lot sizing problems

In this section, we will discuss relevant papers involving Dantzig-Wolfe decomposition and column generation applied to lot sizing problems.

Dantzig-Wolfe decomposition and column generation have been used to find good quality lower bounds for lot sizing problems. Before the seminal paper of Dantzig and Wolfe (1960), Manne (1958) had already implicitly applied the ideas of decomposition for the lot sizing problem with dynamic demand considering several items and capacity constraints. Manne proposed a linear programming model that explicitly considers all possible production schedules. Lambrecht and Vanderveken (1979), Bitran and Matsuo (1986) and Degraeve and Jans (2007) further discuss the formulation proposed by Manne (1958).

Degraeve and Jans (2007) show that the decomposition proposed by Manne, while valid to calculate a strong lower bound, has a structural deficiency when it aims to solve the problem with integrality constraints. The set of feasible solutions for Manne's formulation with integrality constraints is only a subset of feasible solutions for the original integer problem. The main reason for this deficiency is that the solution for the subproblems, i.e. a new column, consists of both setup and production variables and in Manne's formulation the decisions of the setup automatically determine the production quantity decisions according to the Wagner-Whitin property. However, it is very likely that the optimal solution for the capacitated problem will not have this property.

Dzielinski and Gomory (1965) use column generation to handle the formulation with the large number of variables proposed by Manne (1958). Indeed, Manne's formulation is the full master problem obtained when one applies Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) to a formulation with a smaller number of variables. Dzielinski and Gomory (1965) also note that the subproblems that must be solved to generate columns are equivalent to the problem studied by Wagner and Whitin (1958).

Lasdon and Terjung (1971) develop a column generation approach to handle large problems. Algorithms of this type are also addressed by Bahl (1983), Cattrysse et al. (1990), Salomon et al. (1993) and Huisman et al. (2005).

Hindi (1995) presents a heuristic including variable redefinition and column generation. To calculate the upper bound he solves a minimum cost flow problem. Hindi (1996) combines the ideas of linear relaxation, column generation, minimum cost flow network

and Tabu search in a hybrid algorithm. Haase (2005) also solves the lot sizing problem by column generation and finds improved lower bounds.

Considering that both Lagrangian relaxation and Dantzig-Wolfe decomposition have advantages and disadvantages, Huisman et al. (2005) discuss two different ways to combine these two methods in hybrid algorithms to solve the linear relaxation of the master problem. In the first approach they apply Lagrangian relaxation to solve the master problem, i.e., no simplex method is used. In the second approach, they use the simplex method to generate the optimal dual variables of the master problem and the Lagrangian relaxation approach to generate columns. In this latter approach the Lagrangian relaxation is applied to the compact formulation. The ideas are illustrated using the lot sizing problem.

Pimentel et al. (2010) consider the lot sizing problem with setup times and apply the Dantzig-Wolfe decomposition to the classical formulation in two different ways: item decomposition and period decomposition. Furthermore, a third decomposition is presented which applies decomposition per item and period simultaneously. The authors conclude that this last approach provides better lower bounds than those obtained by the other decompositions. They implemented three branch-and-price algorithms to solve the three decomposition models.

Caserta and Voss (2013) also consider the lot sizing problem with setup time and based on the Dantzig-Wolfe approach proposed by Jans and Degraeve (2007), present a math-heuristic approach for the column generation phase of a Dantzig-Wolfe algorithm. The major contribution of this paper lies in the presentation of an algorithm that exploits exact techniques (Dantzig-Wolfe) in a metaheuristic fashion. Their method presents competitive results in terms of both solution quality and running time, but they have performed just few tests (six instances of Trigeiro et al. (1989)) and thus no robust conclusions can be drawn on such a limited test.

Araujo et al. (2015) present a transformed reformulation and valid inequalities that speed up column generation and Lagrangian relaxation for the capacitated lot sizing problem with setup times (CLST) and show theoretically how both ideas are related to dual space reduction techniques. Finally, the authors propose a combination of the two methods proposed by Huisman et al. (2005) in a branch-and-price method. This approach obtains good computational results and avoids the need of a linear programming optimization package.

The aim of this chapter is to find good lower and upper bounds for the single stage problem with unrelated parallel machines extending the ideas proposed in Huisman et al. (2005), Araujo et al. (2015) and Fiorotto and Araujo (2014). Furthermore, different from most other papers that use the traditional decomposition method, which is per item, we

apply decomposition per period and machine.

3.2 Dantzig-Wolfe decomposition applied to the lot sizing problem on parallel machines

In this section, we analyze the theoretical principles of Dantzig-Wolfe decomposition applied to the lot sizing problem on parallel machines. The ideas proposed by Huisman et al. (2005) and Araujo et al. (2015) are used and were extended for the problem being addressed.

In what follows next, we consider the LP relaxation of the Dantzig-Wolfe reformulation.

The decomposition that is commonly used for the lot sizing problem, has as base the formulation (2.1)-(2.5). The capacity constraints (2.4) are the linking constraints and the setup and demand constraints plus the integrality conditions are put into subproblems. Thus, the problem is decomposed into lot sizing problems per item. The approach that will be used in this chapter, takes as base the formulation (2.6)-(2.12) and the linking constraints will be the flow constraints (2.7) and (2.8). The problem is decomposed into independent subproblems per period and per machine containing the capacity and setup constraints plus the integrality conditions. Thus, the extreme points represent production plans for each period and machine. The subproblem solutions are hence production plans indicating for a specific period and machine, which items are produced and in which quantities. These production plans are feasible with respect to the capacity constraints.

Formally, let S_{tj} be the set of all extreme point production plans for period t and machine j. Thus, zt_{tjq} is the new variable representing production plan q for period t on machine j. Note that we assume that the polyhedron is bounded, which is the case for the lot sizing problem because the ending inventory must be zero. Otherwise, a linear combination of extreme rays is also needed. The relationship between the original variables of production and setup and the convex combinations are given by: $zv_{ijtk} = \sum_{q \in S_{tj}} zt_{tjq} zv_{ijtk}$, $\sum_{q \in S_{tj}} zt_{tjq} = 1$, $zt_{tjq} \geq 0$. We also observe that we do not need to redefine the initial inventory variables w_{it} as a convex combination of the extreme points because these variables are not in the constraints (2.9) and (2.10) that will define the subproblems. The resulting relaxed master problem then looks as follows:

$$v_{DWMP} = Min \sum_{i=1}^{n} \sum_{t=1}^{m} ci_{it} w_{it} + \sum_{t=1}^{m} \sum_{j=1}^{r} \sum_{q \in S_{tj}} ct_{tjq} zt_{tjq} \tag{3.1}$$

Subject to :

$$1 \leq \sum_{k=1}^{m}(w_{ik} + \sum_{j=1}^{r}\sum_{q\in S_{1j}} a_{ij1kq} zt_{1jq}) \quad \forall i \in I,\ (\pi_{i1}) \tag{3.2}$$

$$w_{i,t-1} + \sum_{k=1}^{t-1}\sum_{j=1}^{r}\sum_{q\in S_{kj}} a_{ijk,t-1,q} zt_{kjq} \leq \sum_{k=t}^{m}\sum_{j=1}^{r}\sum_{q\in S_{tj}} a_{ijtkq} zt_{tjq} \quad \forall i \in I, t \in T \setminus \{1\},\ (\pi_{it}) \tag{3.3}$$

$$\sum_{q\in S_{tj}} zt_{tjq} = 1 \qquad \forall t \in T, j \in J,\ (\mu_{tj}) \tag{3.4}$$

$$zt_{tjq} \geq 0, w_{it} \geq 0 \tag{3.5}$$

The objective function (3.1) minimizes the total initial inventory cost and the cost of the production plans chosen for each period and machine. The constraints (3.2) and (3.3) are the flow constraints and correspond to the constraints (2.7) and (2.8) with a "smaller than or equal" sign instead of an equality. It means that in each node the sum of the outgoing arcs must be larger than or equal to the sum of the incoming arcs. We use this form because we will solve this master problem with Lagrangian relaxation, and these constraints will be dualized in the objective function with positive dual multipliers p_{it}. Therefore, the inequality interpretation makes it clearer to determine the sign of these relaxed constraints in the Lagrangian objective function. Note that we can make this interpretation since the cost coefficients ci_{it}, sc_{ijt} and cv_{ijtk} in the objective function (2.6) are positive. The constraints (3.4) are the convexity constraints, which guarantee the choice of a convex combination of the extreme points. Note that the flow constraints and the convexity constraints have dual variables called π_{it} and μ_{tj}, respectively.

Due to the huge number of variables a column generation procedure is usually used to solve the master problem. The main idea of this solution method is to start the master problem with some columns (problem called restricted master (RMP)) and progress iteratively generating (with assistance of the subproblems) only the necessary columns until the solution of the restricted master is equal to the solution of the original master problem.

The parameters a_{ijtkq} and ct_{tjq} of the variables zt_{tjq} are defined by the solution of the subproblems. In the subproblems, the objective function minimizes the reduced costs. After rearranging the terms of the objective function, the subproblem for a specific period t and machine j is:

$$z_{SP_{tj}}(\pi,\mu) = Min \sum_{i=1}^{n} sc_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{k=t}^{m-1}(cv_{ijtk} - \pi_{it} + \pi_{i,k+1})zv_{ijtk} + \sum_{i=1}^{n}(cv_{ijtm} - \pi_{it})zv_{ijtm} - \mu_{tj} \quad (3.6)$$

Subject to :

$$\sum_{i=1}^{n} st_{ijt}y_{ijt} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{ijt}sd_{itk}zv_{ijtk} \leq Cap_{jt} \quad (3.7)$$

$$\sum_{k=t}^{m} zv_{ijtk} \leq y_{ijt} \qquad \forall i \in I \quad (3.8)$$

$$y_{ijt} \in \{0,1\}, \quad zv_{ijtk} \geq 0 \qquad \forall i \in I, k \in T, k \geq t \quad (3.9)$$

The constraints of the subproblem (for period t and machine j) are the capacity (3.7) and setup (3.8) constraints plus the constraint (3.9). These subproblems can be solved by the branch-and-bound method proposed by Jans and Degraeve (2004a).

These subproblems are exactly the same as the subproblem (2.19)-(2.22) resulting from the Lagrangian relaxation, except for a constant in the objective function. Indeed, this is always the case if the dualized constraints in the Lagrangian relaxation and the linking constraints of the Dantzig-Wolfe decomposition are the same (Huisman et al., 2005).

Let (y^*_{ijt}, zv^*_{ijtk}) be the optimal solution for a subproblem. A new column is added to the restricted master problem, only if the optimal objective function value of the subproblem ($z_{SP_{tj}}(\pi,\mu)$) is less than zero for at least one pair (j,t). A new column $q \in S_{tj}$ will have the following parameters:

$$a_{ijtkq} = zv^*_{ijtk} \qquad \forall i \in I, \forall k \in T, k \geq t \quad (3.10)$$

$$ct_{tjq} = \sum_{i=1}^{n} sc_{ijt}y^*_{ijt} + \sum_{i=1}^{n}\sum_{k=t}^{m} cv_{ijtk}zv^*_{ijtk} \quad (3.11)$$

Finally, note that the master problem (3.1)-(3.5) is a linear programming problem and can be solved using the simplex method within the column generation procedure.

3.3 Hybrid methods applied to the lot sizing problem on parallel machines

Although the possibility to combine column generation and Lagrangian relaxation has been known for long time, it has only recently been exploited in algorithms. Several

papers from the literature have shown that the combination of these two techniques is a promising tool in the resolution of integer programming problems (Huisman et al., 2005).

It is known that the dual solutions of the master problem obtained when using column generation have a primordial role in the quality of this method. In the standard solution approach using the simplex method for solving the master problem, the optimal dual solution of the restricted master problem is used in the pricing problem. However, it has been pointed out in the literature that optimal dual solutions generated by the simplex algorithm typically cause an unstable behavior of the method. This occurs because these solutions are extreme point solutions and oscillate sharply from one iteration to another, delaying the progress of the method. To counter this behavior, strategies for stabilizing the dual solutions have been proposed in the literature, leading to more efficient variations of the column generation method (see for example, Ben Amor et al. (2007) and Gondzio et al. (2013)).

Following this trend, two different approaches that combine Lagrangian relaxation and Dantzig-Wolfe decomposition are explored in this section.

3.3.1 Lagrangian relaxation applied to the extended formulation (LR/EF)

Instead of using the simplex algorithm to solve the restricted master problem in order to obtain bounds and the optimal dual values of the extended formulation (3.1)-(3.5) (master problem) it is possible to use the Lagrangian relaxation applied to this formulation to approximate these values. The main idea is that using Lagrangian relaxation the cuts made in the dual feasible set (new columns) are deeper than cuts made using the simplex. Therefore, the linking constraints of the master problem (flow constraints, i.e., (3.2)-(3.3)) are transferred to the objective function with the respective dual multipliers. The solutions of the Lagrangian subproblems provide a lower bound to the optimal (LP) relaxation value of the restricted master problem and to get good lower bounds the dual Lagrangian problem must be solved. Cattrysse et al. (1993), Jans and Degraeve (2004a) and Araujo et al. (2015) apply this technique for solving variants of the capacitated lot sizing problem with a single machine.

Formally, to solve approximately the linear programming problem (3.1)-(3.5), the constraints (3.2)-(3.3) are dualized in the objective function (3.1).

$$v_{LRDW}(p) = Min \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} + \sum_{t=1}^{m}\sum_{j=1}^{r}\sum_{q\in S_{tj}} ct_{tjq}zt_{tjq}$$
$$- \sum_{i=1}^{n} p_{i1}(\sum_{k=1}^{m} w_{ik} + \sum_{k=1}^{m}\sum_{j=1}^{r}\sum_{q\in S_{1j}} a_{ij1kq}zt_{1jq} - 1)$$
$$- \sum_{i=1}^{n}\sum_{t=2}^{m} p_{it}(\sum_{k=t}^{m}\sum_{j=1}^{r}\sum_{q\in S_{tj}} a_{ijtkq}zt_{tjq} - w_{i,t-1} - \sum_{k=1}^{t-1}\sum_{j=1}^{r}\sum_{q\in S_{kj}} a_{ijk,t-1,q}zt_{kjq}) \quad (3.12)$$

Subject to :

$$\sum_{q\in S_{tj}} zt_{tjq} = 1 \qquad \forall t \in T, j \in J, \quad (u_{tj}) \quad (3.13)$$

$$zt_{tjq} \geq 0, w_{it} \geq 0 \quad (3.14)$$

The problem (3.12)-(3.14) can be easily solved by inspection. After each iteration of the subgradient optimization method, the multipliers p_{it} are approximations of the optimal dual variables (π_{it}). With this approximation for the vector π_{it} it is possible to calculate an approximation for the vector μ_{tj} that represents the optimal multiplier for the convexity constraints (3.4) (Huisman et al., 2005):

$$u_{tj} = min_{q\in S_{tj}}(ct_{tjq} - \sum_{i=1}^{n}\sum_{k=t}^{m} p_{it}a_{ijtkq} + \sum_{i=1}^{n}\sum_{k=t}^{m-1} p_{i,k+1}a_{ijtkq}) \quad \forall j \in J, \quad (3.15)$$
$$\forall t \in T \setminus \{m\}$$

$$u_{mj} = min_{q\in S_{mj}}(ct_{mjq} - \sum_{i=1}^{n} p_{im}a_{ijmmq}) \qquad \forall j \in J \quad (3.16)$$

The approximated Lagrangian multipliers can be used in the subproblems (3.6)-(3.9) to generate new columns that are added to the restricted master problem and in the next step the optimal dual variables π_{it} and μ_{tj} for the updated restricted master are approximated again by the Lagrangian relaxation.

Lemma 3.1: For each approximation of the solution of the restricted master problem, a lower bound for the master problem can be computed replacing the optimal dual variables by the approximated ones, i.e.:

$$\sum_{t=1}^{m}\sum_{j=1}^{r} z_{SP_{tj}}(p,u) + v_{LRDW}(p) \leq v_{DWMP} \quad (3.17)$$

Proof: This can be proven by starting from the Lagrangian relaxation $v_{LR}(p)$ (2.14), which gives a valid lower bound for any p:

$$
\begin{aligned}
v_{LR}(p) &= \sum_{t=1}^{m}\sum_{j=1}^{r} z_{LR_{tj}}(p) - \sum_{t=1}^{m}\sum_{j=1}^{r} u_{tj} + \sum_{t=1}^{m}\sum_{j=1}^{r} u_{tj} + \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} \\
&- \sum_{i=1}^{n} p_{i1}(\sum_{k=1}^{m} w_{ik} - 1) - \sum_{i=1}^{n}\sum_{t=2}^{m} p_{it}(-w_{i,t-1}) \\
&= \sum_{t=1}^{m}\sum_{j=1}^{r} z_{SP_{tj}}(p,u) + \sum_{t=1}^{m}\sum_{j=1}^{r} u_{tj} + \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} \\
&- \sum_{i=1}^{n} p_{i1}(\sum_{k=1}^{m} w_{ik} - 1) - \sum_{i=1}^{n}\sum_{t=2}^{m} p_{it}(-w_{i,t-1}) \\
&= \sum_{t=1}^{m}\sum_{j=1}^{r} z_{SP_{tj}}(p,u) + \sum_{t=1}^{m-1}\sum_{j=1}^{r} min_{q\in S_{tj}}(ct_{tjq} - \sum_{i=1}^{n}\sum_{k=t}^{m} p_{it}a_{ijtkq} \\
&+ \sum_{i=1}^{n}\sum_{k=t}^{m-1} p_{i,k+1}a_{ijtkq}) + \sum_{j=1}^{r} min_{q\in S_{mj}}(ct_{mjq} - \sum_{i=1}^{n} p_{im}a_{ijmmq}) \\
&+ \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} - \sum_{i=1}^{n} p_{i1}(\sum_{k=1}^{m} w_{ik} - 1) - \sum_{i=1}^{n}\sum_{t=2}^{m} p_{it}(-w_{i,t-1}) \\
&= \sum_{t=1}^{m}\sum_{j=1}^{r} z_{SP_{tj}}(p,u) + v_{LRDW}(p) \ \blacksquare
\end{aligned}
$$

The advantage of approximating the optimal dual variables by the Lagrangian relaxation, is that in the case of alternative dual solutions, column generation algorithms tend to converge more quickly using dual variables produced by interior point methods than with extreme point dual variables calculated by the simplex method (Bixby et al., 1992; Barnhart et al., 1998). From this perspective, the Lagrangian multipliers can provide a better representation and speed up the convergence. Computational experiments performed in Jans and Degraeve (2004a) indicate that the use of the Lagrangian multipliers indeed speeds up the convergence and reduces the problem of degeneration. The Lagrangian relaxation also has the additional advantages that during the subgradient method, feasible solutions are possibly generated. Furthermore, the subgradient update is quick and easy to implement and finally, this procedure eliminates the need for a commercial LP optimizer.

Algorithm 1 shows the application of Lagrangian relaxation to the extended formulation in order to obtain a lower and upper bound:

Algorithm 1: Lagrangian Relaxation Applied to the Extended Formulation

Input: Initial RMP; multipliers; $max.\ iteration$.

Output: Lower bound (LB); upper bound (UB).

1 Let $LB = -\infty$, $UB = +\infty$, $p_{it} = u_{tj} = 0$;

2 While $(z_{SP_{tj}}(p,u) < 0)$ do:

3 *Lagrangian step counter* := 0;
4 Do while (*Lagrangian step counter*< *max. iteration*)
5 Solve the problem $v_{LRDW}(p)$ (3.12)-(3.14);
6 Update the multipliers p_{it} with the subgradient method;
7 *Lagrangian step counter* := *Lagrangian step counter* +1;
8 end(do while);
9 Use best approximation for the Lagrangian multipliers of π_{it} and calculate an approximation of μ_{tj} (u_{tj}) with the formula (3.15)-(3.16);
10 Solve the subproblems (3.6)-(3.9) with the approximate dual prices p_{it} and u_{tj};
11 Apply a feasibility heuristic (see Section 3.4);
12 Update the bounds LB and UB;
13 If (*subproblem value* < 0) then add columns;
14 end (while)

3.3.2 Lagrangian relaxation applied to the extended and compact formulations ($LR/EF/CF$)

Following the ideas proposed in Araujo et al. (2015), this approach is based on the observation that when the Lagrangian relaxation is obtained by dualizing exactly those constraints that are the linking constraints in the Dantzig-Wolfe decomposition, the same subproblem results. Indeed, the subproblem to calculate the minimum reduced cost in the Dantzig-Wolfe decomposition given by (3.6)-(3.9) and the Lagrangian subproblem (2.19)-(2.22) are the same except for a constant in the objective function. Consequently, the solutions generated by the Lagrangian relaxation (on the compact formulation) can be used to add new columns in the restricted master problem. In $LR/EF/CF$, we combine the approach of Section 3.3.1 with the ideas discussed above in this section.

The main idea of this combined method ($LR/EF/CF$) is to use the Lagrangian relaxation on the extended formulation (master problem) to approximate the dual variables of the *RMP* (as explained in Section 3.3.1), and afterwards use these variables as initial dual variables in a column generation procedure based on Lagrangian relaxation of the compact formulation (original formulation).

Therefore, with this method would like to achieve the follows objectives: to avoid an unstable behavior in the column generation method (degeneration); converge quickly by adding blocks of columns generated by Lagrangian relaxation.

This method is represented in the Algorithm 2. After generating some initial columns,

the Lagrangian relaxation of the extended formulation problem (3.12)-(3.14) is solved a first time with the subgradient optimization method that provides an approximation p_{it} and u_{tj} for π_{it} and μ_{tj}. Then, the subproblems (3.6)-(3.9) are solved. If all the reduced cost to generate columns are positive the procedure is stopped and a lower bound can be found (using the formula (3.17)). Otherwise, it moves to Lagrangian relaxation of the compact formulation. The initial Lagrangian multipliers are equal to the current dual prices found by the solution of the problem (3.6)-(3.9) and the Lagrangian inner loop starts. The Lagrangian problem $(2.14) - (2.18)$ is solved and the result provides a lower bound. The inner loop of Lagrangian iterations continues (and we check if we reached the maximum value). In each step new multipliers p_{it} are obtained with the subgradient optimization method and the Lagrangian problem is solved with the current dual prices and a new lower bound is obtained. If this value is better than the current lower bound, we update the current lower bound. Then, for each period t we check if we can find a column with negative reduced cost. This provides a new column because the Lagrangian subproblem $(2.19) - (2.22)$ is identical to the subproblem to generate columns (3.6)-(3.9). The reduced cost of each column should be checked using the dual prices of the last time that the restricted master problem was solved. After a fixed number of Lagrangian inner loops, if none of the generated column enters in the restricted master problem, the best approximation according to the predefined parameters, for the optimal solution is found. Otherwise, the columns with negative reduced costs are added to the master problem. Next, we optimize again the restricted master problem with the new added columns using Lagrangian relaxation. After a predefined number of iterations of the subgradient optimization method we return to Lagrangian relaxation of the compact formulation using the new dual prices generated by the approximate solution of the master problem. The procedure is stopped when no columns price out.

Algorithm 2: Lagrangian Relaxation Applied to the Extended and Compact Formulations

Input: Initial PMR; multipliers; $max.\ iteration$; $max.\ it\_Lagrangian$

Output: Lower bound (LB); upper bound (UB).

1 Let $LB = -\infty$, $UB = +\infty$, $p_{it} = u_{tj} = 0$;
2 While $(z_{SP_{tj}}(p, u) < 0)$ do:
3 $Lagrangian\ step\ counter := 0$;
4 Do while ($Lagrangian\ step\ counter < max.\ iteration$)
5 Solve the problem $v_{LRDW}(p)$ (3.12)-(3.14);
6 Update the multipliers p_{it} with the subgradient method;
7 $Lagrangian\ step\ counter := Lagrangian\ step\ counter + 1$;
8 end(do while);

Use best approximation for the Lagrangian multipliers of π_{it} and calculate an approximation of μ_{tj} (u_{tj}) with the formula (3.15)-(3.16);
Solve the subproblems (3.6)-(3.9) with the approximate dual prices p_{it} and u_{tj};
Apply a feasibility heuristic (see Section 3.4) and update the upper bound (UB);
If ($z_{SP_{tj}}(p,u) < 0$) Then
  $it\_Lagrangian = 0$;
  Do while ($it\_Lagrangian \leq max.\, it\_Lagrangian$)
    Calculate the Lagrangian bound ($v_{LR}(p)$) (2.14) and update the lower bound (LB);
    Update the Lagrangian multipliers p_{it} using subgradient;
    $it\_Lagrangian := it\_Lagrangian + 1$;
    Add columns if $z_{LR_{tj}}(p)(2.19) < 0$ and not added yet;
  end (do while);
 end (if);
end (while);

3.4 Extended primal heuristic

3.4.1 General overview

To obtain a feasible solution (upper bound) we extended the feasibility heuristic proposed by Fiorotto and Araujo (2014) (Section 2.3.2) by adding an initialization and an improvement stage and by making several changes in the backward and forward stages. The main difference in the backward stage is that we apply this stage taking into account the inventory levels, and thus we avoid needing to reapply this stage after the forward stage. For the forward stage the main difference is that we determine the order that the items are analyzed and the machines to produce.

In general, the new feasibility heuristic steps can be described as follows:

Extended Primal Heuristic Steps

- Stage 1: **Backward Stage (free up capacity)**
  For each item, check if total production is in excess of total demand. To do this find the total inventory quantity for each period and check if the inventory of the last period is greater than zero. If it is, remove excess production.

- Stage 2: **Forward Stage**
  Starting with the first period and moving to the last, we check for each item if the demand is satisfied. If it is not, try to make the solution feasible in the following order:
  - 2.1: Produce the quantity still needed with machines already set up (and checking the machines in increasing order of production cost) in the current period and if not possible, in previous periods;
  - 2.2: Produce the remaining quantity by performing a new machine setup checking the machines in increasing order of average unit production cost in the current period or if not possible, in previous periods;
  - 2.3: If, after all the above attempts, demand is still not met for a given item i' in period t, we check among previously-made-feasible items for one which has inventory left at the end of period t and remove the corresponding excess production, thus freeing up capacity. Try to satisfy the demand for the given item i', by using this capacity.

- Stage 3: **Improvement Stage**
  After applying the Backward and Forward Stage ($B\&F$) for all Lagrangian iterations that improved the lower bound, we obtain (in most cases) a feasible production plan, i.e., a setup plan and according production quantities. Since these production quantities are determined by the $B\&F$ heuristic, they are not necessarily the optimal production plans for the given setup plan. Then, for the best found solution, we check if with the same setup schedule, we can find better production quantities than suggested by the $B\&F$ heuristic. We fix the setup variables to their current value, as given by the $B\&F$ heuristic solution, and solve the remaining LP with an optimization package to determine the optimal according production quantities.

Following, each stage is described in detail.

3.4.2 Initialization

We would like to determine for each item which is the cheapest machine to produce on. This is not trivial, since we have to take into account both the unit production cost and the setup cost. The total cost, however, depends on the proposed production plan. In order to establish a ranking, different from Fiorotto and Araujo (2014), we calculate an approximate average unit cost per item and per machine, assuming that we use the Economic Order Quantity (EOQ) as production quantity (Andriolo et al., 2014). The EOQ is a classic model of inventory management, in which the idea is to determine the

optimal number of units to order so that we minimize total inventory holding costs and ordering costs. Figure 3.1 illustrates a EOQ production plan. Note that the new order is delivered in completely when inventory reaches zero.

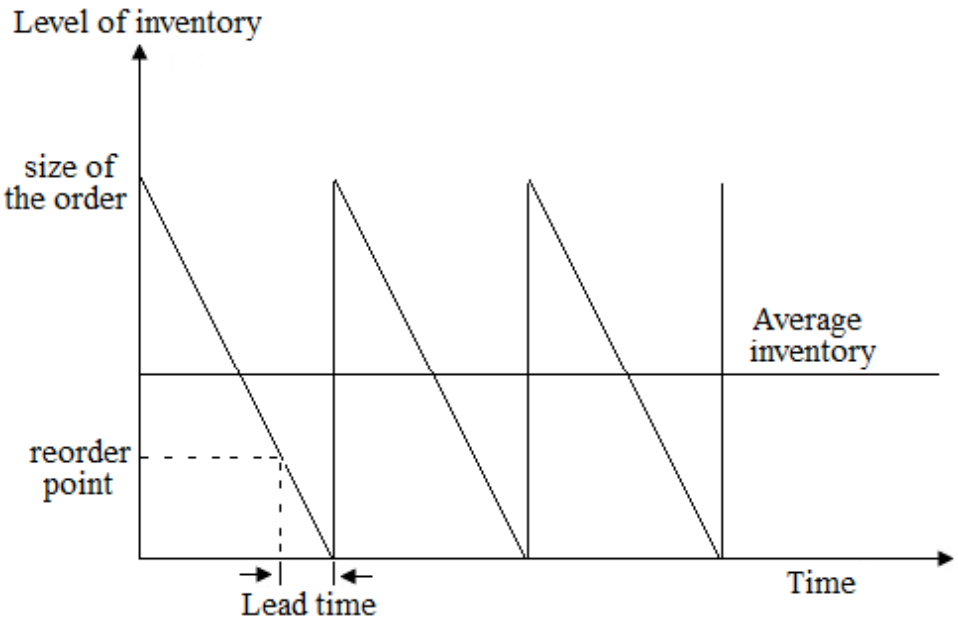


Figure 3.1: Graphic of the EOQ.

We also observe that the EOQ can be applied only in a setting with constant demand. Therefore, we first calculate the average demand per period for each item:

$$AVD(i) = \frac{\sum_{t=1}^{m} d_{it}}{m}, \quad i = 1, 2, ..., n$$

After this, we calculate the EOQ for each item and machine, i.e., $EOQ(i,j) = \sqrt{\frac{2 \times AVD(i) \times sc_{ijt}}{hc_{it}}}$ $\forall i \in I, j \in J$. Note that in the data sets sc_{ijt}, hc_{it}, st_{ijt}, vt_{ijt}, and Cap_{jt} are time invariant. To deal with the discrete time horizon, when the EOQ is too small, in order to avoid needing to do several orders in the same period to meet demand, we set $EOQ(i,j) := AVD(i)$ if $EOQ(i,j) \leq AVD(i)$.

We need to consider two separate cases to calculate the approximate average unit cost (AC_{EOQ}) for each item and machine. The two cases consider the relationship between EOQ and the available capacity:

I) If $EOQ(i,j) \leq \frac{(Cap_{jt} - st_{ijt})}{vt_{ijt}}$, then $AC_{EOQ}(i,j) = \frac{sc_{ijt}}{EOQ(i,j)} + vc_{ijt}$;

II) However, if $EOQ(i,j) > \frac{(Cap_{jt} - st_{ijt})}{vt_{ijt}}$, then $Q := \frac{(Cap_{jt} - st_{ijt})}{vt_{ijt}}$ and $AC_{EOQ}(i,j) = \frac{sc_{ijt}}{Q} + vc_{ijt}$;

The idea behind the formula is to estimate the average production cost for an item on each machine taking into account both the unit production cost and the setup cost.

The values for $AC_{EOQ}(i,j)$ establishes the production priority order of the machines for each item.

3.4.3 Backward stage

This first stage to make a solution feasible consists of freeing up capacity if we have excess production, and to do this, the production quantities for each item are analyzed and removed if necessary. Initially we calculate the inventory of each item at the end of each period:

$$\Delta(i,t) := \sum_{l=1}^{t}\sum_{j=1}^{r} x_{ijl} - \sum_{l=1}^{t} d_{il} \;\; \forall i \in I, t \in T$$
$$If \;\; \Delta(i,t) < 0 \;\; Then \;\; \Delta(i,t) := 0$$

If the inventory at the end of the horizon is strictly positive, i.e., if $\Delta(i,m) > 0$ for some $i = 1, 2, ..., n$ then the excess production of this item must be eliminated. The aim is to remove the excess inventory at the end of the horizon by reducing the production quantities in the earliest possible period, without creating any (additional) backlog. The reason we look for the earliest period is that freeing up capacity in earlier periods provides more flexibility than freeing up capacity in later periods.

To do this, starting with the last period m and following the sequence $t = m, m-1, ..., 1$, find the first period with zero inventory, indicated by t'. Then we calculate the total quantity that should be removed in the period $t'+1$. To do this, we determine the $min\{\Delta(i,t)\}$ in the interval $t'+1, t'+2, ..., m$. Let t^* be the period for which we have this minimum.

Denoting this minimum by $\Delta(i,t^*)$, remove this amount from the production quantity in the period $t'+1$. Look for a machine j with the greatest (AC_{EOQ}) and then do the following analysis.

If $min\{\Delta(i,t^*), x_{ij,t'+1}\} = x_{ij,t'+1}$ remove the total amount produced and consequently the setup for this machine and period. Thus, the remaining capacity is updated:

$$Cap'_{j,t'+1} := Cap'_{j,t'+1} + x_{ij,t'+1} \times vt_{ij,t'+1} + st_{ij,t'+1}$$

The values of $\Delta(i,t^*)$, the inventory level and the variables are also updated by:

$$\Delta(i,t) := \Delta(i,t) - x_{ij,t'+1}, t = t'+1, ..., m;$$
$$x_{ij,t'+1} := 0; y_{ij,t'+1} := 0;$$

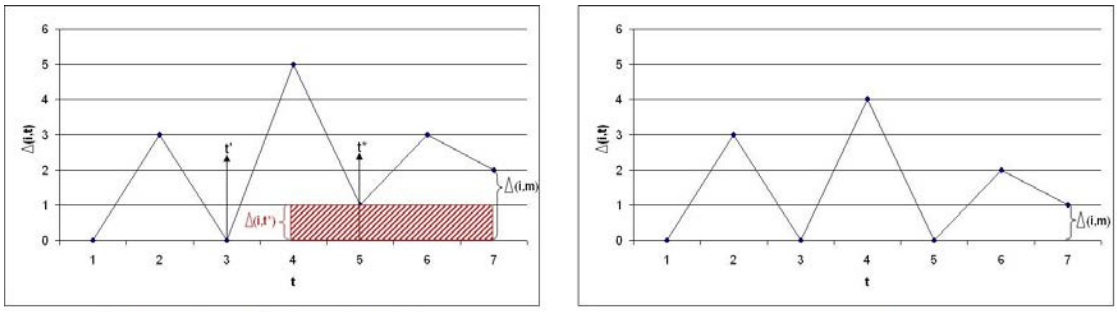
Then move on to the other machines to remove any remaining excess production.

However, if $Min\{x_{ij,t'+1}, \Delta(i,t^*)\} = \Delta(i,t^*)$, remove the total amount that can be removed in this period and recalculate the inventory level and production quantity:

$$Cap'_{j,t'+1} := Cap'_{j,t'+1} + \Delta(i,t^*) \times vt_{ij,t'+1};$$
$$\Delta(i,t) := \Delta(i,t) - \Delta(i,t^*), t = t'+1, ..., m;$$
$$x_{ij,t'+1} := x_{ij,t'+1} - \Delta(i,t^*);$$

If after this step the inventory in the last period is equal to zero, we have eliminated the production excess of this item. However, if the inventory of the last period is still greater than zero we do the same analysis again, i.e., starting with the last period m, we find the first period t with zero inventory.

In the Figure 3.2, we provide an example with seven periods. In part (a) we see that there is excess inventory at the end of the horizon. Note that starting with the last period, the first period with zero inventory (t') is the period 3 and the minimum inventory $\Delta(i,t^*)$ is 1 in period 5 (t^*). In part (b) we see the inventory quantity for all periods after removing 1 unit from production in period 4. Note that the inventory of the last period is still greater than zero, therefore we have to do the same analysis again.



(a) Inventory quantity in each period

(b) Inventory quantity after reducing excess

Figure 3.2: Example of the backward stage.

3.4.4 Forward stage

When the process of eliminating the excess production is over, we start to check if the demand is met and try to satisfy demand for pairs of items and periods for which demand is currently not met. For this, the analysis will be done from period 1 through to the last period ($t = 1, 2, ..., m$).

First, we determine the order in which the items are analyzed according to the opportunity cost ($OC(i)$) of each item using the decreasing order. We calculate $OC(i) = AC_{EOQ}(i, j') - AC_{EOQ}(i, \bar{j})$, where j' and $\bar{j}$ are the most expensive and cheapest machine respectively for item i. The idea behind this is that if the opportunity cost value is high, it is very important to produce this item on the cheapest machine so we will try to make this item feasible first. For other hand, if the opportunity cost value is low we can produce the item on any machine because the difference is not very significant.

To check if demand is met, an array is needed that counts the total production for each item up to the given period. Initially $\Omega(i) = 0 \ \forall i \in I$. Starting from period $t = 1$, go through all the items (according to the order calculated) calculating $\Omega(i) := \Omega(i) + \sum_{j=1}^{r} x_{ijt}$ and do the following analysis.

If $\Omega(i) \geq d_{it}$ the demand for (i, t) is met. Simply remove this specific demand from the total production, in other words, $\Omega(i) := \Omega(i) - d_{it}$ and move on to analyze the next item. Observe that in this case we have $\Omega(i) \geq 0$.

If $\Omega(i) < d_{it}$, demand is not met for this pair (i, t), then the feasibility process needs to be started. The quantity of items that need to be produced is easily calculated:

$$\Phi = d_{it} - \Omega(i)$$

Once this amount has been calculated, we have three attempts to make the solutions feasible (stages 2.1, 2.2 and 2.3 in the overview of the heuristic in Section 3.4.1).

• **Attempt 2.1**

Start at the current period t and go back period by period to the first period ($\tau = t, t-1, ...1$) checking first if the machines already setup for this item ($y_{ij\tau} = 1$) have sufficient capacity to produce the quantity that is still needed to satisfy demand.

We check over all periods $\tau = t, t-1, ...1$ and within each time period we search over all machines (on which a setup is done) according to the increasing order of the value of unit production cost. Note that here we use the unit production cost instead AC_{EOQ} because the setup is already done.

Therefore, if $y_{ij\tau} = 1$, that is, if the machine is already set up, we calculate the maximum amount that can be produced with the remaining capacity, and compare this to the amount we still need to produce. We calculate the minimum of these two values:

$$Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\}$$

Then, if $Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\} = \Phi$ the machine can produce the required amount to satisfy demand for this pair (i,t). Therefore, the values are updated by:

$$Cap'_{j\tau} := Cap'_{j\tau} - \Phi \times vt_{ij\tau};\ \ x_{ij\tau} := x_{ij\tau} + \Phi;\ \ \Phi := 0;\ \ \Omega(i) := 0$$

However, if $Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\} = \frac{Cap'_{j\tau}}{vt_{ij\tau}}$, there is still a quantity which needs to be produced. So we update the values:

$$Cap'_{j\tau} := 0;\ \ x_{ij\tau} := x_{ij\tau} + \frac{Cap'_{j\tau}}{vt_{ij\tau}};\ \ \Phi := \Phi - \frac{Cap'_{j\tau}}{vt_{ij\tau}}$$

Next we check the next machine in the list or, if we have checked all the machines, move back to the previous period and repeat the same analysis.

• **Attempt 2.2**

If after checking all previous periods and machines with a setup, there is still some demand which is not met, we check the possibility of adding a setup to another machine but initially only if it can produce all of the remaining quantity required to satisfy demand. Again starting from $\tau = t$ to $\tau = 1$ and checking the machines on which no setup is done for item i (according to the order calculated by AC_{EOQ}) if there is a period τ and a machine j such that $Cap'_{j\tau} \geq \Phi \times vt_{ij\tau} + st_{ij\tau}$ then adjust the production accordingly:

$$x_{ij\tau} := x_{ij\tau} + \Phi;\ \ y_{ij\tau} := 1;\ \ Cap'_{j\tau} := Cap'_{j\tau} - (\Phi \times vt_{ij\tau}) - st_{ij\tau};\ \ \Omega(i) := 0$$

However, if there is still no single machine with enough capacity to produce the total missing demand, we repeat the loop and open up the possibility of setting up several machines. At this stage, the criteria used to determine the order of which machine to set up is again the order calculated according to the average unit production cost (AC_{EOQ}).

• **Attempt 2.3**

If after all the previous attempts, it is still not possible to meet the demand for a given item (i) in a given period (t), look among previous items that have already been made feasible for one that has inventory at the end of period t, in other words, one for which the value of omega is positive. If it is found, make the necessary adjustments, i.e., remove the excess production of the found item (freeing up capacity) and try to produce the required amount of item (i). Thus, we search in the following order $\tau = t, t-1, ...1$ and within each time period we search over all items according to the decreasing order calculated by AC_{EOQ} and then:

If $\Omega(i') > 0$ and $y_{i'j\tau} = 1$, in other words, an item with a quantity in inventory and a machine and period already setup for this item was found, do the following analysis:

If $\Omega(i') \geq x_{i'j\tau}$ then, as well as freeing up capacity, a setup of the machine can be removed i.e.:

$$Cap'_{j\tau} := Cap'_{j\tau} + x_{i'j\tau} \times vt_{i'j\tau} + st_{i'j\tau};\ \ y_{i'j\tau} := 0;\ \ \Omega(i') := \Omega(i') - x_{i'j\tau};\ \ x_{i'j\tau} := 0$$

However, if $\Omega(i') < x_{i'j\tau}$, then just free up the capacity:

$$Cap'_{j\tau} := Cap'_{j\tau} + x_{i'j\tau} \times vt_{i'j\tau};\ \ x_{i'j\tau} := x_{i'j\tau} - \Omega(i');\ \ \Omega(i') := 0$$

With the capacity freed, calculate the total quantity of items (i) that can be produced and check again to see if it is possible to produce the required amount to meet demand in the following way:

If the machine is already set up for the item i, then calculate:

$$Min\{\Phi, \frac{Cap'_{j\tau}}{vt_{ij\tau}}\}$$

However, if there is no setup for item i, calculate:

$$Min\{\Phi, \frac{(Cap'_{j\tau} - st_{ij\tau})}{vt_{ij\tau}}\}$$

Then, if the minimum is Φ, a quantity sufficient to satisfy the demand for this pair (i, τ) can be produced and so we adjust the production amount accordingly:

$$Cap'_{j\tau} := Cap'_{j\tau} - \Phi \times vt_{ij\tau} - st_{ij\tau};\ \ x_{ij\tau} := x_{ij\tau} + \Phi,\ \ y_{ij\tau} = 1$$

If the machine was already set up, we do not need to subtract the setup time.

However, if the minimum is $\frac{Cap'_{j\tau}}{vt_{ij\tau}}$ or $\frac{(Cap'_{j\tau} - st_{ij\tau})}{vt_{ij\tau}}$, there is still a quantity left to be produced and so we add the quantity that can be produced, for example, if the machine is already set up for item i:

$$Cap'_{j\tau} := Cap'_{j\tau} - \frac{Cap'_{j\tau}}{vt_{ij\tau}} \times vt_{ij\tau};\ \ x_{ij\tau} := x_{ij\tau} + \frac{Cap'_{j\tau}}{vt_{ij\tau}}$$

If the machine is not set up for item i we have to use $\frac{(Cap'_{j\tau} - st_{ij\tau})}{vt_{ij\tau}}$ instead $\frac{Cap'_{j\tau}}{vt_{ij\tau}}$. After that, we move on to the next item (already made feasible) with a quantity in inventory and check the same conditions.

Once these feasibility stages have been performed for all items, move on to the next period of the forward stage and start the analysis again, for each item from first to last, trying to make the solution feasible. Repeat this procedure until the last period of the forward stage.

The overall pseudo code for this forward stage is as follows. To simplify the reading we will denote by $i := 1, ..., n$ the decreasing order of the items according to the opportunity cost, $j := 1, ..., r$ the increasing order of the machines according to the value of production cost and $k := 1, ..., r$ the increasing order of the machines according to the value of AC_{EOQ}. All the values of $\Omega(i)$ are initialized at zero.

for $t := 1, ..., m$ **do**

for $i := 1, ..., n$ **do**
  for $j := 1, ..., r$ **do**
    $\Omega(i) := \Omega(i) + x_{ijt}$
  end
  if $(\Omega(i) > d_{it})$ **then**
    $\Omega(i) := \Omega(i) - d_{it}$
  else
    $\Phi := d_{it} - \Omega(i)$
    for $\tau := t, t-1, ..., 1$
      for $j := 1, ..., r$
        if $(y_{ij\tau} = 1$ and $\Phi > 0)$ **then**
          Calculate the whole amount that can be produced on this machine in this period, updating of the necessary parameters;
        end
      end
    end
    for $\tau := t, t-1, ..., 1$
      for $k := 1, ..., r$
        if $(y_{ik\tau} = 0$ and $\Phi > 0)$ **then**
          First try to find a machine that can produce all quantity to meet demand. If we can not find, we go back to this loop and open the possibility to make more than one setup, updating of the necessary parameters;
        end
      end
    end
    for $\tau := t, t-1, ..., 1$
      for $i' := 1, ..., i-1$
        for $k := r, ..., 1$
          if $(\Phi > 0, \Omega(i') > 0$ and $y_{i'k\tau} = 1)$ **then**
            Remove the excess inventory and try to produce the required amount of item i, updating of the parameters;
          end
        end
      end
    end

end (else)
end (for i)
end (for j)

Finally, at the end of the forward stage, we calculate the objective function value of the solution, if the solution becomes feasible, i.e. if all the demand is met.

3.4.5 Improvement stage

This last stage consists of trying to improve the objective function value of the solution. We fix the best setup plan found and solve the remaining LP problem using a optimization package solver, i.e., if we denote by y^*_{ijt} the value of the setup variables found by the heuristic, we have to solve the following problem:

$$Min \sum_{i=1}^{n}\sum_{t=1}^{m} ci_{it}w_{it} + \sum_{i=1}^{n}\sum_{j=1}^{r}\sum_{t=1}^{m}\sum_{k=t}^{m} cv_{ijtk}zv_{ijtk} \tag{3.18}$$

Subject to :

$$1 = \sum_{k=1}^{m} w_{ik} + \sum_{j=1}^{r}\sum_{k=1}^{m} zv_{ij1k} \qquad \forall i \in I \tag{3.19}$$

$$w_{i,t-1} + \sum_{j=1}^{r}\sum_{k=1}^{t-1} zv_{ijk,t-1} = \sum_{j=1}^{r}\sum_{k=t}^{m} zv_{ijtk} \qquad \forall i \in I, t \in T/\{1\} \tag{3.20}$$

$$\sum_{k=t}^{m} zv_{ijtk} \leq y^*_{ijt} \qquad \forall i \in I, j \in J, t \in T \tag{3.21}$$

$$\sum_{i=1}^{n}\sum_{k=t}^{m} vt_{ijt}sd_{itk}zv_{ijtk} \leq Cap_{jt} - \sum_{i=1}^{n} st_{ijt}y^*_{ijt} \qquad \forall j \in J, t \in T \tag{3.22}$$

$$zv_{ijtk} \geq 0, \quad w_{it} \geq 0 \qquad \forall i \in I, j \in J, t \in T \ \ \forall k \in T, k \geq t \tag{3.23}$$

Finally, we remove possible setups that are not used. We check for all the constraints (3.21) if there is a setup ($y^*_{ijt} = 1$) and $\sum_{k=t}^{m} zv_{ijtk} = 0$, and remove the setup if this is the case.

Note that the heuristic is applied to the solutions obtained by the Lagrangian subproblems. Consequently, the quality of the solution of the heuristic is strongly linked to quality of the production plans determined by these subproblems.

3.5 Computational results

The algorithms described in the previous sections were tested on a total of 2160 instances proposed in Toledo and Armentano (2006). The classes and parameters are gen-

erated in the same way as in the Chapter 2 and so will be omitted.

We compare the hybrid methods described in Sections 3.3.1 and 3.3.2 (LR/EF and $LR/EF/CF$) with other existing methods: 1) the Lagrangian heuristic proposed in Toledo and Armentano (2006) (per item decomposition), here denoted by TA, 2) the Lagrangian heuristic applied to the compact formulation (2.6)-(2.12) as proposed in Fiorotto and Araujo (2014) (using the period and machine decomposition) denoted by LR/CF and 3) the CPLEX 12.6 software package, applied to the formulation (2.6)-(2.12). The tests were done on a personal computer Intel Core-I5, 2.27GHz with 6Gb of RAM and the Windows 7 operating system.

The parameters used in the computational tests for the method LR/EF, described in Section 3.3.1, which solves the master problem with Lagrangian relaxation are as follows: 900 iterations of the subgradient optimization method; initialization of the dual prices to zero; size of the initial step of the subgradient method equal to 1 which is decreased by multiplying the latest step size by a factor of 0.7 if the Lagrangian bound does not improve in the last 10 iterations. For the method $LR/EF/CF$, described in Section 3.3.2, we use the same parameter setting described in the above method to approximate the solution of the master problem with the Lagrangian relaxation. In addition, the columns are generated with Lagrangian relaxation and in this case we use the following settings: 200 iterations of the subgradient optimization method; initialization of the dual prices according to those obtained in the solution of the master problem; size of the initial step equal to 1 which is decreased by multiplying the latest step size by a factor of 0.6 if the Lagrangian bound does not improve in the last iteration. After 5 iterations of the column generation procedure, the number of iterations of the subgradient optimization method is reduced from 200 to 1, this basically means that we just do regular column generation for that point on, without subgradient optimization. For the method LR/CF proposed in Fiorotto and Araujo (2014): they start the subgradient optimization method fixing the dual variables to zero; the size of the initial step is equal to 1 and decreases by multiplying by 0.6 if the Lagrangian solution is not improved in the last 50 iterations; 2500 iterations are made in total. For the method TA proposed in Toledo and Armentano (2006): the size of the initial step is equal to 1.75 and divided by 2 if the Lagrangian solution is not improved in the last 25 iterations; 150 iterations are made in total.

In the tables in which we compare the gaps, the upper bounds of the two methods proposed in this chapter are obtained by the heuristic described in Section 3.4. Moreover, the gap is calculated using the formula $gap = \frac{(UB-LB)*100}{LB}$, in which UB and LB denote the upper and lower bounds found by each respective solution approach. Note that the best results are highlighted in bold numbers.

Table 3.1 shows the overall performance of the several methods (computational times

in seconds, the gaps and the percentage of Better (B)/Equal (E)/Worse (W) solutions in terms of gap that the proposed methods found compared to CPLEX) aggregated over the 8 combinations of the following three factors: capacity, setup cost and setup time. Each row of the table represent thus the average of 270 instances generated for each class. Moreover, for CPLEX we fixed the time limit for each instance to the same time as used by the method $LR/EF/CF$. The results from TA are taken directly from their paper and the computational experiments were done on AMD Athlon XP2600 with 2.08GHz and 1GB of RAM. We note that despite the higher computation times of the hybrid methods, mainly the method $LR/EF/CF$, they present better gaps than the Lagrangian heuristics TA and LR/CF proposed in Toledo and Armentano (2006) and Fiorotto and Araujo (2014), respectively. Note that this difference increases significantly when the problems with high setup cost (HS) are considered. Furthermore, for these instances the proposed method $LR/EF/CF$ found better gaps compared to CPLEX for more than 50% of the instances. In most cases, we have the same conclusion comparing the results found by the hybrid methods with CPLEX.

	CPLEX		TA		LR/CF		LR/EF			LR/EF/CF		
Class	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	B/E/W	T(s)	Gap	B/E/W	T(s)
NLL	1.7	81.8	2.0	3.9	1.5	3.3	1.4	36/31/33	70.1	**1.3**	41/27/32	81.8
TLL	**1.8**	71.5	3.4	8.9	2.4	4.1	2.3	27/29/44	35.2	1.9	30/38/32	71.5
NHL	5.8	20.8	11.7	5.3	8.9	4.9	5.8	28/48/24	16.9	**4.6**	56/23/21	20.8
THL	7.6	17.3	18.5	7.8	12.6	6.5	6.1	45/21/34	25.7	**5.7**	58/18/24	17.3
NLH	3.2	68.5	2.5	4.9	1.7	3.5	1.7	39/33/28	76.7	**1.5**	48/24/28	68.5
TLH	2.8	86.1	4.2	14.4	2.9	4.7	2.6	37/30/33	27.7	**2.0**	45/21/34	86.1
NHH	6.9	23.2	13.8	5.9	9.9	5.2	5.9	44/19/37	17.3	**4.9**	59/16/25	23.2
THH	8.1	17.1	23.5	8.4	13.9	6.6	6.2	63/22/15	20.9	**5.7**	74/12/14	17.1

Table 3.1: General average gap and computational times for each class.

In Tables 3.2 to 3.4 we analyze the results aggregated per periods, items and machines considering all classes.

Note that for Table 3.2 and the following tables, we are not able to give the results for TA, since the results were not provided according to this classification in the paper of Toledo and Armentano (2006).

Table 3.2 shows that both the gaps and the computation times increase with an increasing number of periods. On the other hand, Table 3.3 shows that while the computations times increase with an increasing number of items, the gaps decrease. This was also observed by Trigeiro et al. (1989) for the single machine lot sizing problem with setup times. From Table 3.4 we can conclude that with the decomposition methods, the gaps increase but the computation times decrease with an increasing number of machines. In Tables 3.2 to 3.4, the new hybrid method always give average gaps that are much better compared to the other decomposition methods. This is also the case when we compare the results

to CPLEX, except for the case with 25 items in Table 3.3 and the case with 2 machines in Table 3.4 for which CPLEX provides slightly better gaps. Similar conclusion can be made when we compare the percentage of instances for which the proposed methods found better solutions than CPLEX.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Periods	Gap	T(s)	Gap	T(s)	Gap	B/E/W	T(s)	Gap	B/E/W	T(s)
6	4.2	11.2	4.7	2.8	2.9	43/32/25	7.8	**2.4**	55/27/18	11.2
12	4.3	39.3	6.6	6.7	4.2	36/27/37	29.1	**3.8**	48/20/32	39.3
18	5.7	93.9	8.9	13.5	4.9	41/28/31	72.1	**4.2**	50/22/28	93.9

Table 3.2: General average gap and computational times aggregated per period.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Items	Gap	T(s)	Gap	T(s)	Gap	B/E/W	T(s)	Gap	B/E/W	T(s)
6	9.1	6.9	11.3	2.9	7.1	51/35/14	5.0	**6.4**	62/26/12	6.9
12	3.9	23.9	6.4	6.0	3.4	39/30/31	15.9	**2.7**	54/22/24	23.9
25	**1.2**	113.2	2.5	14.1	1.5	30/22/48	88.1	1.3	37/21/42	113.2

Table 3.3: General average gap and computational times aggregated per item.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Machines	Gap	T(s)	Gap	T(s)	Gap	B/E/W	T(s)	Gap	B/E/W	T(s)
2	**2.4**	69.1	4.7	8.2	3.2	28/29/43	57.8	2.6	34/25/41	69.1
4	5.5	41.2	6.9	6.8	3.8	48/28/24	29.7	**3.3**	61/22/17	41.2
6	6.3	33.88	8.6	4.5	5.0	44/30/26	21.6	**4.5**	58/22/20	33.8

Table 3.4: General average gap and computational times aggregated per machine.

In Tables 3.5 to 3.8 we set for each class the lower bound found by the linear relaxation to 100%, and calculated the other values relative to this. For example, for the first class in Table 3.5, the linear relaxation after the branching is 0.34% better than the linear relaxation.

Table 3.5 presents the lower bounds found by CPLEX for each class 1) from linear relaxation at the root node (L. Rel. column); 2) after applying the cuts (cuts column); and 3) after branching (branch. column). These lower bounds are compared with those generated by all methods that are being analyzed. Note that the proposed hybrid methods found, in all cases, better lower bounds than the TA and LR/CF methods. The lower bounds (column LB) found using the hybrid methods are on average better than the lower bounds found by CPLEX after branching, especially for the problem with high setup costs. Furthermore, the results of the hybrid methods are also better with respect to the percentage of instances resulting in improved lower bounds compared to CPLEX (column B/E/W). This means that, if the lower bounds generated by these methods were used in the root node of CPLEX, in addition to getting better gaps, it could also helps the solver to prune more in the solution tree and find better feasible solutions.

In relation to the two proposed methods in this chapter, it is observed that the differences are small, however, the method $LR/EF/FC$ obtained a small advantage in all instances.

	CPLEX (LB)			TA	LR/CF	LR/EF		LR/EF/FC	
Class	L. Rel.	Cuts	Branch.	LB	LB	LB	B/E/W	LB	B/E/W
NLL	100	100.27	100.34	99.94	100.34	100.67	40/38/22	**100.84**	42/39/19
TLL	100	100.42	100.54	99.78	100.46	101.29	38/44/18	**101.30**	38/45/17
NHL	100	101.29	101.55	99.73	100.79	104.12	63/32/5	**104.33**	63/32/5
THL	100	101.39	101.81	99.34	101.87	105.66	61/36/3	**105.81**	62/35/3
NLH	100	100.26	100.35	99.93	100.39	100.90	46/31/23	**100.96**	48/30/22
TLH	100	100.38	100.52	99.70	100.54	101.19	43/37/20	**102.05**	45/35/20
NHH	100	101.43	101.73	99.54	100.82	104.99	69/27/4	**105.15**	69/27/4
THH	100	101.42	102.13	98.44	102.76	106.46	66/32/2	**107.13**	67/31/2

Table 3.5: General average lower bounds for each class.

Analyzing the lower bound results separately per period, item and machine (Tables 3.6, 3.7 and 3.8), we conclude that the hybrid methods found on average better lower bounds in all configurations. The difference is particularly pronounced for the instances with a small number of items or a high number of machines. Observe that in these cases, the percentage of instances in which CPLEX found a better lower bound is very low. On the other hand, the difference in lower bounds is small for the instances with a large number of items or a small number of machines. The number of periods has a less pronounced impact on the lower bounds, even though here we observe again that the hybrid methods provide much better lower bounds.

	CPLEX (LB)			TA	LR/CF	LR/EF		LR/EF/FC	
Periods	L. Rel.	Cuts	Branch.	LB	LB	LB	B/E/W	LB	B/E/W
6	100	101.12	101.50	99.55	100.93	102.58	49/33/18	**102.78**	50/33/17
12	100	100.75	100.87	99.45	100.89	103.16	54/34/12	**103.36**	55/34/11
18	100	100.58	100.62	99.65	101.15	103.69	56/38/6	**103.89**	57/38/5

Table 3.6: General average lower bounds aggregated per period.

	CPLEX (LB)			TA	LR/CF	LR/EF		LR/EF/FC	
Items	L. Rel.	Cuts	Branch.	LB	LB	LB	B/E/W	LB	B/E/W
6	100	102.09	102.68	98.91	102.04	106.29	65/31/4	**106.63**	65/31/4
12	100	100.40	100.54	99.81	100.73	102.66	56/37/7	**102.83**	57/37/6
25	100	100.08	100.14	99.93	100.20	100.48	38/37/25	**100.56**	40/37/23

Table 3.7: General average lower bounds aggregated per item.

In the Tables 3.9 to 3.12 we compare the upper bounds. We fixed the upper bound found by CPLEX to 100%, and calculated again the upper bounds found by the other methods relative to this. Table 3.9 shows the heuristic's behavior (upper bounds) for all classes. The results show that we improved the heuristic proposed by Fiorotto and Araujo

	CPLEX (LB)			TA	LR/CF	LR/EF		LR/EF/FC	
Machines	L. Rel.	Cuts	Branch.	LB	LB	LB	B/E/W	LB	B/E/W
2	100	100.47	101.05	99.45	100.77	101.70	36/39/25	**101.88**	38/40/22
4	100	100.71	100.82	99.62	100.99	103.68	60/32/8	**103.81**	61/31/8
6	100	100.95	100.98	99.61	101.95	103.95	63/34/3	**104.09**	63/34/3

Table 3.8: General average lower bounds aggregated per machine.

(2014) considering that we found better upper bounds for all classes. Note that these improvements are bigger for the classes with high setup cost. For 6 out of the 8 classes, we also improved on average CPLEX upper bounds, whereas for the two remaining classes the increase is only 0.45% and 0.20%. We observe that for many classes, CPLEX produces better upper bounds for more instances (indicated by the fact that W is larger than B in the columns B/E/W) if a feasible solution is found, even though the average upper bound is better for the best new heuristic. The high percentage of feasible solutions ($\%FS$) found by the hybrid methods show the efficiency of these heuristic. CPLEX, however, did not find feasible solutions for a considerable number of instances within the time limit imposed. Instances with high setup costs seem specifically difficult for CPLEX.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Class	UB	%FS	UB	%FS	UB	%FS	B/E/W	UB	%FS	B/E/W
NLL	100	92.22	100.79	100.00	100.20	100.00	26/16/58	**99.99**	100.00	32/31/37
TLL	**100**	92.96	101.52	100.00	100.72	100.00	21/10/69	100.45	100.00	29/16/55
NHL	100	82.96	101.97	100.00	100.29	100.00	16/7/77	**99.33**	100.00	38/7/55
THL	100	69.25	99.23	99.62	97.04	100.00	15/5/80	**96.11**	99.62	33/5/62
NLH	100	95.92	99.15	100.00	98.72	99.25	28/9/63	**98.37**	100.00	41/19/40
TLH	100	86.29	99.66	100.00	98.86	98.88	31/10/59	**98.60**	100.00	43/18/39
NHH	100	70.00	101.84	100.00	100.23	100.00	18/3/79	**99.11**	100.00	36/3/61
THH	**100**	80.00	104.34	99.25	101.25	99.62	23/1/76	100.20	99.62	30/3/67

Table 3.9: General average upper bounds for each class.

Finally, Tables 3.10, 3.11 and 3.12 compare the upper bounds considering the number of periods, items and machines. We note that the new heuristic is better and the differences are bigger for problems with 6 periods, 6 items and 4 or 6 machines.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Periods	UB	%FS	UB	%FS	UB	%FS	B/E/W	UB	%FS	B/E/W
6	100	94.16	99.27	100.00	98.65	99.72	25/10/65	**97.74**	100.00	41/20/39
12	100	76.25	100.97	99.72	99.93	99.86	22/8/70	**99.55**	99.86	35/14/51
18	100	78.33	102.53	99.72	100.65	99.44	19/6/75	**99.98**	99.72	29/5/66

Table 3.10: General average upper bounds aggregated per period.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Items	UB	%FS	UB	%FS	UB	%FS	B/E/W	UB	%FS	B/E/W
6	100	65.27	100.57	99.44	98.22	99.58	27/11/62	**97.01**	99.44	43/21/36
12	100	87.50	101.46	100.00	100.28	99.44	20/7/73	**99.64**	100.00	37/14/49
25	**100**	96.11	101.57	100.00	100.53	100.00	19/6/75	100.43	100.00	25/4/71

Table 3.11: General average upper bounds aggregated per item.

	CPLEX		LR/CF		LR/EF			LR/EF/CF		
Machines	UB	%FS	UB	%FS	UB	%FS	B/E/W	UB	%FS	B/E/W
2	**100**	97.91	101.66	99.46	100.79	99.30	17/5/78	100.51	99.44	25/2/73
4	100	80.83	100.97	100.00	99.52	100.00	23/10/67	**98.80**	100.00	39/14/47
6	100	70.00	100.25	100.00	98.03	100.00	26/9/65	**97.26**	100.00	41/20/39

Table 3.12: General average upper bounds aggregated per machine.

3.6 Conclusion

In this chapter, the lot sizing problem with capacity constraints and parallel machines was studied. A reformulation of the problem using the variable redefinition approach proposed by Eppen and Martin (1987) was used. Based on the literature, two hybrid solution methods that combine Lagrangian relaxation and Dantzig-Wolfe decomposition were extended to the problem considered. In both methods, the Dantzig-Wolfe decomposition is applied using as linking constraints the flow constraints. The problem is hence decomposed per period and per machine instead of the classical per item decomposition. A feasibility heuristic based on production transfers in order to satisfy the demand constraints is extended. This strategy was compared to two Lagrangian heuristics proposed in the literature and also compared to CPLEX 12.6. The computational results show that the proposed hybrid methods are efficient with respect to both the lower bounds and upper bounds compared the existing methods.

Chapter 4

An analysis of formulations for the capacitated lot sizing problem with setup crossover

In this chapter we study the lot sizing problem with setup crossover which is an extension of the standard big bucket capacitated lot sizing problem (CLSP).[1] The general idea is that the first setup operation of each planning period can already start in the previous period, if not all the capacity is used in that previous period. This provides more flexibility in the planning and increases the possibility of finding feasible and better solutions compared to the standard assumption. Two different formulations have been presented in the literature to model a setup crossover. Since these formulations have not been compared directly to each other, we present a computational study to determine which is the best formulation. Furthermore, we explore ideas to avoid the necessity of defining new extra binary variables to model the setup crossover and propose symmetry breaking constraints for both formulations from the literature. Finally, we quantify the value of this type of flexibility in a computational experiment and analyse which factors influence this value.

4.1 Introduction

Over the past decades, there has been an increasing interest in the application of these models, and researchers have been able to incorporate more and more real world features into lot sizing problems.

The problem considered in this chapter is the single stage, single machine, multi-

[1] A compact version of this chapter has been published as a research report *(CIRRELT-2014-57)* and submitted to an international journal.

product, big bucket lot sizing problem with setup times. Several different products can be produced in the same time period on the same machine. A setup must be done for each type of product that is produced in a specific period. In the standard version of this problem (Trigeiro et al., 1989) the setup for the first product type produced in a period starts at the beginning of that period (see Figure 4.1a on page 80). In this chapter we study an extension of this lot sizing problem that includes the possibility of a setup crossover. The idea is that in certain cases setup operations can be interrupted at the end of a period and resumed at the beginning of the next period, in other words, the setups can span over two periods. This implies that the first setup in period t can already start at the end of period $t-1$ if there is some capacity left, and continue at the beginning of period t (see Figure 4.1b on page 80). This flexibility can result in more efficient solutions compared to the standard assumption (where the setup time is restricted to be contained within the period) since we free up capacity in period t by moving (partially) the setup of the first product to the previous period. In the big bucket models, the setup times are smaller than or equal to the capacity.

It is important to note the differences between the concepts of setup crossover and setup carryover. While with setup crossover the setups can span over two periods, the setup carryover allows a setup state to be maintained from one period to the next one, in other words, if we finish a period t producing a particular item i it is possible to start the period $t+1$ producing the item i without performing a new setup for this item.

Although setup crossover is a natural extension of the standard assumption, just a few studies have considered it, due to the difficulty in dealing with the underlying problems (Mohan et al., 2012; Belo-Filho et al., 2014). All the studies that handle setup crossovers in their formulations have added extra binary variables to the formulations indicating if there is a setup crossover in a period or not, which increases the difficulty of the formulations.

The aim of this chapter is: 1) to propose new ideas to avoid the necessity of defining new extra binary variables to model the setup crossover; 2) to propose new constraints to break the symmetry which is present in the formulations from the literature; 3) to compare the two formulations proposed in the literature to determine which formulation is the best; 4) to analyse the impact of the proposed adaptations of these formulations (i.e. no binary variables and symmetry breaking constraints), and 5) to determine the value of the flexibility provided by the setup crossover and analyse the factors that have an impact on this value.

4.2 Literature review on lot sizing problem with setup crossover

There is a vast amount of literature on big-bucket capacitated lot sizing problems (CLSP) with setup times, where setup times have to be contained completely within one period (see e.g. Trigeiro et al. (1989)). These models have been extended to deal with various industrial issues (see Jans and Degraeve (2008) for an overview), including setup carryover and setup crossover.

Several papers analyze the extension with setup carryover. Sox and Gao (1999) propose two formulations for the CLSP with setup carryover. The first one extends the formulation proposed by Trigeiro et al. (1989) and the second one uses the shortest path reformulation and the ideas proposed by Eppen and Martin (1987). Suerie and Stadtler (2003) propose a formulation for the CLSP with setup carryover based on the simple plant location (Krarup and Bilde, 1977) and their computational tests have shown that this formulation is better than the formulations proposed by Sox and Gao (1999). Wu and Shi (2011) present a theoretical and computational study about the relationship of four different reformulations of this problem. The authors conclude that the reformulations yield a similar performance. Gopalakrishnan et al. (2001) develop a tabu search heuristic to solve the CLSP with setup carryover and using the data sets from Trigeiro et al. (1989) they compute the effectiveness of the setup carryover. Their results indicate an 8% reduction in total cost on average through setup carryover compared with the standard CLSP.

Regarding the problem with setup crossover for the small bucket problem, Suerie (2006) studies the lot sizing and scheduling problem and proposes two formulations that correctly handle setup crossovers which allow "long" setup times (i.e. setup times can be bigger than the capacity in one period). The author compares his results with the results found by the standard lot sizing and scheduling problem and concludes that the proposed formulations produce more feasible and improved solutions. Kaczmarczyk (2009) proposed two new mixed integer programming models based on the proportional lot sizing problem with setup crossover and long setup times. The results proved that the proposed models are easier to solve (using standard MIP methods) and showed a better performance over the literature, mainly for problem in which the setup times are longer than the period length.

For the big bucket problem, Sung and Maravelias (2008) present a mixed-integer programming formulation for the capacitated lot sizing problem allowing setup carryover and crossover (CLSP-SCC). The authors consider sequence independent setups, non-uniform time periods and long setup times. They show in a detailed way how to deal with the boundary of the periods using setup crossover with the assumption that the setup cost is

accounted for at the beginning of the setup. Finally they discuss how their formulation can be extended for problems with idle time, parallel units, families of products, backlog and lost sales.

Menezes et al. (2010) propose a formulation for the CLSP-SCC considering sequence-dependent and non-triangular setups, allowing subtours and enforcing minimum lot sizing. They propose two lemmas to demonstrate that their formulation is more efficient than the classical lot sizing and scheduling problem. Moreover, they present an example that shows the improvement of the solutions allowing setup crossover compared to the classical formulation.

Kopanos et al. (2011) develop a formulation for the CLSP-SCC with backlog where the items are classified into families. The approach considers that the setups are family sequence-dependent, and sequence-independent for items belonging to the same family. The formulation is tested for a complex real world problem in the continuous bottling stage of a beer production facility and it finds good solutions for problems with hundreds of items.

Mohan et al. (2012) include the possibility of setup crossover for the formulation proposed by Suerie and Stadtler (2003) that handles the problem with setup carryover and compare the improvement obtained by adding the crossover in the formulation with setup carryover. They conclude that in nine out of fifteen problem instances tested, their formulation yielded better solutions or removed infeasibility.

Camargo et al. (2012) propose three formulations for the two-stage lot sizing and scheduling problem and one of these considers setup crossover, which is achieved by a continuous-time representation. From the computational results, they conclude that despite delivering the worst performance in terms of CPU times, the formulation with setup crossover is the most flexible of the three to incorporate setup-related features.

Belo-Filho et al. (2014) consider the problem CLSP-SCC with backlog. They propose two formulations for the problem, the first one is built on top of the formulation of Sung and Maravelias (2008) and the second one proposes a time index disaggregation, defining the start and the completion time periods of the setup operation. They show the relationship between the proposed formulations and compare their formulations with the formulation proposed by Sung and Maravelias (2008). Finally they point out that setup crossover is an important modeling feature in case setup times consume a considerable part of the period capacity.

4.3 Mathematical models

In this section, to contextualize we first present the classical formulation and the simple plant location reformulation (Krarup and Bilde, 1977) for standard problem (without crossover). Next, we present two formulations for the problem with setup crossover based on the ideas proposed by Menezes et al. (2010) and Mohan et al. (2012) and three different ways to model the setup crossover without defining new extra binary variables. Finally, we propose new constraints for the formulations proposed by Menezes et al. (2010) and Mohan et al. (2012) to break the symmetry resulting from the presence of alternative optimal solutions.

Note that we will not consider the possibility of initial inventory in the formulations of this chapter because we will use at the computational experiments the data sets proposed by Trigeiro et al. (1989) and it is well known that these data sets are all feasible.

4.3.1 Classical models

We first present the classical formulation of the capacitated lot sizing problem. This formulation is based on the formulation of Trigeiro et al. (1989).

For the mathematical formulation of the problem consider the following data:

Parameters

$I = \{1, ..., n\}$: set of items;
$T = \{1, ..., m\}$: set of periods;
d_{it}: demand of item i in period t;
$sd_{it\tau}$: the sum of the demand for item i, from period t until period τ ($\tau \geq t$);
hc_{it}: unit inventory cost of item i in period t;
sc_{it}: setup cost for item i in period t;
vc_{it}: production cost of item i in period t;
st_{it}: setup time for item i in period t;
vt_{it}: production time of item i in period t;
Cap_t: capacity (in units of time) in period t;

Decision variables

x_{it}: number of units produced of item i in period t;
s_{it}: quantity of inventory of item i at the end of period t;
y_{it}: binary setup variable, indicating the production or not of item i in period t;

- Classical formulation (CF)

$$v(CF) = Min \sum_{i=1}^{n} \sum_{t=1}^{m} (sc_{it} y_{it} + vc_{it} x_{it} + hc_{it} s_{it}) \tag{4.1}$$

Subject to:

$$s_{i,t-1} + x_{it} = d_{it} + s_{it} \qquad \forall i \in I, t \in T \tag{4.2}$$

$$\sum_{i=1}^{n} (st_{it} y_{it} + vt_{it} x_{it}) \leq Cap_t \qquad \forall t \in T \tag{4.3}$$

$$x_{it} \leq min\{(Cap_t - st_{it})/vt_{it}, sd_{itm}\} y_{it} \qquad \forall i \in It \in T \tag{4.4}$$

$$y_{it} \in \{0,1\},\ x_{it} \geq 0,\ s_{it} \geq 0,\ s_{i0} \geq 0,\ s_{im} = 0 \qquad \forall i \in I, t \in T \tag{4.5}$$

The objective function (4.1) minimizes the total setup, production and inventory costs. The constraints (4.2) guarantee the inventory balance in each period. Next are capacity limits (4.3) and the machine setup constraints (4.4). In order to make the formulation stronger, we limit the production for each item in constraints (4.4) by both the remaining demand and the maximum possible production with the available capacity. Finally, constraints (4.5) define the variables domains.

Various research papers have used alternative formulations to model the classical lot sizing formulation. Two important reformulations have been proposed. A first one deals with the reformulation of the problem as a shortest path problem in which a redefinition of the variables proposed by Eppen and Martin (1987) is the strategy used (Fiorotto and Araujo, 2014) and discussed in the Chapter 2. A second one consists of a reformulation based on the Simple Plant Location problem studied in Krarup and Bilde (1977). Various theoretical and computational results concerning such reformulations have been published in the literature. Considering that the authors that have studied the lot sizing problem with setup crossover used in their papers the simple plan location reformulation, and after performing some preliminary computational tests we have chosen to use the simple plant location reformulation for all formulations presented on this chapter.

Next we present the simple plant location reformulation of the problem (4.1)-(4.5). For the reformulation the following parameters are defined:

cs_{itk}: total production and holding cost for producing one unit of item i in period t to satisfy demand of period k:

$$cs_{itk} = (vc_{it} + \sum_{u=t}^{k-1} hc_{iu}) d_{ik}.$$

There are also the following new variables for the model:

x_{itk}: fraction of the demand for item i in period k produced in period t.

The reformulation based on the simple plant location is as follows:

- Simple plant location reformulation ($F0$)

$$v(F0) = Min \sum_{i=1}^{n}\sum_{t=1}^{m} sc_{it}y_{it} + \sum_{i=1}^{n}\sum_{t=1}^{m}\sum_{k=t}^{m} cs_{itk}x_{itk} \tag{4.6}$$

Subject to:

$$\sum_{k=1}^{t} x_{ikt} = 1 \qquad \forall i \in I, t \in T \mid d_{it} > 0 \tag{4.7}$$

$$\sum_{i=1}^{n} st_{it}y_{it} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{it}d_{ik}x_{itk} \leq Cap_t \qquad \forall t \in T \tag{4.8}$$

$$x_{itk} \leq y_{it} \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.9}$$

$$y_{it} \in \{0,1\},\ x_{itk} \geq 0 \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.10}$$

The objective function (4.6) minimizes the total cost, which consists of the setup cost and the aggregated production and holding costs. The constraints (4.7) ensure that demand is met for each period. The capacity constraints (4.8) limit the sum of the total setup and production times. The setup constraints (4.9) do not allow any production in period t unless a setup is done. Finally, constraints (4.10) define the variables domains.

4.3.2 Models proposed in literature for the problem with setup crossover

In this section we present the formulations of the lot sizing problem with setup crossover proposed in literature. These formulations are based on the formulations of Menezes et al. (2010) and Mohan et al. (2012). Both papers also include the possibility of setup carryover. We present here the way the setup crossover is formulated in these papers, without considering the setup carryover extensions. There are other papers in the literature for extensions of the CLSP with setup crossover as discussed in the literature review. However, for these formulations the ways to model the setup crossover are similar to that of the papers previously mentioned, and therefore they will be omitted.

Before presenting the formulation, we need to define some new variables:

Decision variables

v_{it}: binary variable, indicating if the setup is split between period t and period $t+1$ for item i;

u_t: extra time borrowed in period t for the setup in period $t+1$.

The first mathematical formulation based on the ideas proposed by Menezes et al. (2010) is as follows:

- Model adding extra binary variables ($F1$)

$$v(F1) = Min \sum_{i=1}^{n}\sum_{t=1}^{m} sc_{it}y_{it} + \sum_{i=1}^{n}\sum_{t=1}^{m}\sum_{k=t}^{m} cs_{itk}x_{itk} \tag{4.11}$$

Subject to:

$$\sum_{k=1}^{t} x_{ikt} = 1 \qquad \forall i \in I, t \in T \mid d_{it} > 0 \tag{4.12}$$

$$\sum_{i=1}^{n} st_{it}y_{it} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{it}d_{ik}x_{itk} + u_t \leq Cap_t + u_{t-1} \qquad \forall t \in T \tag{4.13}$$

$$x_{itk} \leq y_{it} \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.14}$$

$$u_{t-1} \leq \sum_{i=1}^{n} v_{i,t-1}st_{it} \qquad \forall t \in T \tag{4.15}$$

$$v_{i,t-1} \leq y_{it} \qquad \forall i \in I, t \in T \tag{4.16}$$

$$\sum_{i=1}^{n} v_{i,t-1} \leq 1 \qquad \forall t \in T \tag{4.17}$$

$$y_{it} \in \{0,1\},\ v_{i,t-1} \in \{0,1\},\ v_{i0} = 0,\ u_{t-1} \geq 0,\ u_0 = 0 \quad \forall i \in I, t \in T \tag{4.18}$$

$$x_{itk} \geq 0 \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.19}$$

The objective function (4.11) minimizes the total cost. Constraints (4.12) guarantee that the demand is satisfied in each period. The capacity constraints (4.13) limit the sum of the total setup times and production times, considering the time borrowed from the previous period and the time added to the next period in case of setup crossover. The setup constraints (4.14) do not allow any production in period t unless a setup is done. Constraint (4.15) limits the borrowed time in period $t-1$ to be used in period t to the value of the setup time of the product for which we allow the crossover. We cannot have a crossover from period $t-1$ to period t if there is no setup in period t, as imposed by constraint (4.16). Constraints (4.17) state that the setup can be split across periods for at most one item and finally, the conditions (4.18) and (4.19) on the variables complete the formulation.

The second formulation is based on the ideas proposed by Mohan et al. (2012) and the main difference is the way to limit the time for the setup crossover (constraints (4.15) and (4.16) of the previous formulation). Before presenting the formulation, we need to define new variables:

New decision variables:

z_{it}: 1 if a complete setup is done in period t for item i, 0 otherwise;

$l_{i,t-1}$: extra time borrowed in period t to start the setup of the item i;

f_{it}: total time used to complete the setup of the item i in period t;
v_{it}: 1 if setup crossover between period $t-1$ and period t for item i with splits being $l_{i,t-1}$ and f_{it}, respectively (that is $f_{it} + l_{i,t-1} = st_{it}$).

The second formulation is then as follows:

- Model separating complete setups ($F2$)

$$v(F2) = Min \sum_{i=1}^{n}\sum_{t=1}^{m}(sc_{it}z_{it} + sc_{it}v_{it}) + \sum_{i=1}^{n}\sum_{t=1}^{m}\sum_{k=t}^{m} cs_{itk}x_{itk} \tag{4.20}$$

Subject to:

$$\sum_{k=1}^{t} x_{ikt} = 1 \qquad \forall i \in I, t \in T \mid d_{it} > 0 \tag{4.21}$$

$$\sum_{i=1}^{n} st_{it}z_{it} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{it}d_{ik}x_{itk} + \sum_{i=1}^{n} l_{it} + \sum_{i=1}^{n} f_{it} \leq Cap_t \qquad \forall t \in T \tag{4.22}$$

$$x_{itk} \leq z_{it} + v_{it} \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.23}$$

$$f_{it} + l_{i,t-1} = v_{it}st_{it} \qquad \forall i \in I, t \in T \tag{4.24}$$

$$\sum_{i=1}^{n} v_{it} \leq 1 \qquad \forall t \in T \tag{4.25}$$

$$y_{it} \in \{0,1\},\ v_{it} \in \{0,1\},\ l_{it} \geq 0,\ l_{i0} = 0,\ f_{it} \geq 0 \qquad \forall i \in I, t \in T \tag{4.26}$$

$$x_{itk} \geq 0 \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.27}$$

The objective function (4.20) minimizes the total setup, production and inventory costs. The constraints (4.21) guarantee that the demand is satisfied in each period. Constraints (4.22) ensure that the total capacity consumed during a period for production and setups is less than or equal to the available capacity. The setup constraints (4.23) do not allow any production in period t unless a setup is done (either complete or crossover). When a setup is split, constraints (4.24) ensure that the split times add up to the total setup time. Constraints (4.25) state that the setup can be split across periods for at most one item. Finally, constraints (4.26) and (4.27) define the variable domains.

4.3.3 Proposed models

The first new proposed model formulates a restricted version of the problem. It is built upon the idea that it is possible to limit the quantity of borrowed time (u_t) in each period by the lowest value of the setup times. Although this restricts the set of feasible solution, it avoids the necessity of defining extra binary variables for the setup crossover. Note that this formulation will have a higher (or equal) optimal objective function value

compared to formulations $F1$ and $F2$ (as we will formally discuss in Section 4.4.1) and hence the solution can be used as a heuristic. This new formulation is defined as follows:

- Model with minimum setup time ($F3$)

$$v(F3) = Min \sum_{i=1}^{n}\sum_{t=1}^{m} sc_{it}y_{it} + \sum_{i=1}^{n}\sum_{t=1}^{m}\sum_{k=t}^{m} cs_{itk}x_{itk} \tag{4.28}$$

Subject to:

$$\sum_{k=1}^{t} x_{ikt} = 1 \qquad \forall i \in I, t \in T \mid d_{it} > 0 \tag{4.29}$$

$$\sum_{i=1}^{n} st_{it}y_{it} + \sum_{i=1}^{n}\sum_{k=t}^{m} vt_{it}d_{ik}x_{itk} + u_t \leq Cap_t + u_{t-1} \qquad \forall t \in T \tag{4.30}$$

$$x_{itk} \leq y_{it} \qquad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.31}$$

$$u_{t-1} \leq \min_{\forall j \in I}\{st_{jt}\} \qquad \forall t \in T \tag{4.32}$$

$$y_{it} \in \{0,1\},\ x_{itk} \geq 0,\ u_{t-1} \geq 0,\ u_0 = 0 \quad \forall i \in I, t \in T, k \in T, k \geq t \tag{4.33}$$

The objective function (4.28) and the constraints (4.29), (4.30) and (4.31) are the same as constraints (4.11), (4.12), (4.13) and (4.14) in the formulation $F1$. Constraints (4.32) limit the extra time allowed for a setup crossover to the lowest setup time of all products. The last constraints (4.33) state the domain of the variables.

- Model with minimum active setup time ($F4$)

The second proposed formulation is an extension of the first one and thus, it also formulates a restricted version of the problem. The idea is that we can only use extra capacity from period $t-1$ for a product that is setup in period t, i.e., we can limit the borrowed extra time (in period $t-1$) to the minimum time of the active setups in period t.

The following modification of the constraints (4.32) handles this extension:

$$u_{t-1} \leq st_{it}y_{it} + \max_{\forall j \in I}\{st_{jt}\}(1 - y_{it}) \qquad \forall i \in I, t \in T \tag{4.34}$$

Constraints (4.34) limit the extra time allowed for a setup crossover to the minimum setup time of the active setups. If the setup is not active in period t for product i, i.e. $y_{it} = 0$, then the extra time is limited by the maximum setup time. Otherwise, if the setup is active for product i, i.e. $y_{it} = 1$, then the extra time is limited by exactly this

setup time. As we have this constraint for every item, we are limiting the u_{t-1} variables to the minimum active setup time.

The second proposed formulation, $F4$, is the same as $F3$ with constraints (4.32) replaced by the constraints (4.34). This formulation will also result in a higher (or equal) optimal objective function value when compared to formulations $F1$ and $F2$, and hence the solution can only be used as a heuristic.

- Model dropping the extra binary variables ($F5$)

The third new formulation is based on formulation $F1$. Analyzing the constraints (4.15) to (4.17) we observe that the integrality constraints on v_{it} can be dropped. The idea is that it is always feasible to limit the allowable time for a setup crossover to the maximum of the active setup times in a period (as formally discussed in Section 4.4.1). If less time is allowed (because there is not enough idle capacity in the previous period) or needed, the u_t variables can always assume a lower value. This constraint is still imposed if we drop the integrality constraints on the v_{it} variables (assuming positive setup times). The right-hand side of (4.15) cannot be more than the maximum of the active setup times because of constraints (4.16) and (4.17) together even if the binary conditions on the v_{it} variables are dropped. Note that when the binary decisions are dropped, the variables v_{it} does not necessarily indicate anymore which item is split, since it can assume fractional values. They are only used to determine the maximum time allowed for the crossover. The formulation $F5$ consists of the objective function (4.11), subject to constraints (4.12)-(4.19) with the integrality constraints on v_{it} dropped.

- Model to break the symmetry of formulation $F1$ ($F1'$)

For formulation $F1$, it is possible that alternative (optimal) solutions exists with the same (optimal) objective function value, as will be formally explained in Section 4.4.1. The problem with these alternative or symmetric solutions is that they can increase the total computation time needed due to duplication in the branch-and-bound tree (see e.g. Sherali and Smith (2001), Jans (2009) and Jans and Desrosiers (2013)). We can exclude these alternative solutions by explicitly imposing that the item with the highest active setup time in period $t+1$ is always chosen as the item for which we have a setup crossover between periods t and $t+1$. This is always feasible since the variable u_t can take a value which is lower than this setup time, or can even take the value of zero (if no idle capacity is available in period t, or if a setup crossover is not beneficial). To impose this condition, we first have to order the items in a decreasing order of their setup times. Next we have to add the following symmetry breaking constraints to formulation $F1$:

$$v_{1,t-1} = y_{1t} \qquad \forall t \in T \setminus \{1\} \tag{4.35}$$

$$v_{i,t-1} \geq y_{it} - \sum_{j=1}^{i-1} y_{jt} \qquad \forall i \in I \setminus \{1\}, \forall t \in T \setminus \{1\} \tag{4.36}$$

Constraints (4.35) and (4.36) impose in each period the setup crossover for the product with the highest active setup time. Note that as the items are ordered according to the decreasing order of setup time, item 1 has the highest setup time. Therefore, constraint (4.35) enforces the setup crossover between periods $t-1$ and t for the first item (i.e. the one with the highest setup time) only if this item is setup in period t. Constraint (4.36) enforces a setup crossover between periods $t-1$ and t for item i only if this item is setup in period t and if none of the items with a higher setup time have been setup in period t. Note that constraint (4.16) still prevents a crossover for an item if there is no setup. Formulation $F1$ augmented with constraints (4.35) and (4.36) will be called $F1'$.

- Model to break the symmetry of formulation $F2$ ($F2'$)

For formulation $F2$, we observe as well that there can be several alternative solutions with the same objective function value (see Lemma 4.3 in Section 4.4.1). The reason is basically the same as for formulation $F1$.

We also have proposed a type of symmetry breaking constraint to formulation $F2$ to impose the setup crossover always for the product with the highest active setup time. As in the previous formulation, we first have to order the items according to a decreasing order of setup times and then we add to formulation $F2$ the following new constraints:

$$\sum_{j=1}^{i-1} v_{jt} \geq z_{it} \qquad \forall t \in T, i \in I \setminus \{1\} \tag{4.37}$$

Constraints (4.37) imposes that if there is a full setup for item i, this means there must have been a partial setup for an item $j < i$ (assuming an decreasing order of setup time). Constraints (4.37) will hence assign the setup crossover to the item with the highest active setup time.

4.4 Analysis of the formulations

4.4.1 Theoretical analysis of the formulations

In this section we prove the relationship among the optimal objective function values for all of the discussed formulations and we show that we can always construct an

alternative solution with the same objective function value imposing the setup crossover for the product with the highest active setup time. Note that $v(F)$ indicates the optimal objective function value of formulation F and $S(F)$ denotes the set of feasible solutions.

Lemma 4.1 $v(F0) \geq v(F3) \geq v(F4) \geq v(F5) = v(F1) = v(F2)$.

Proof: The first inequality ($v(F0) \geq v(F3)$) is trivial, since by adding the setup crossover the flexibility is increased and better solutions can be found. Moreover, it is clear that $S(0) \subseteq S(3)$ because by fixing $u_t = 0\ \forall t \in T$ in formulation $F3$ we obtain the classical formulation $F0$.

Formulation $F3$ is more restrictive than formulation $F4$. Comparing the right-hand side of constraints (4.32) and (4.34) we have that $\min_{\forall j \in I}\{st_{jt}\} \leq st_{it}y_{it} + \max_{\forall j \in I}\{st_{jt}\}(1 - y_{it}), \forall i \in I$. Therefore $S(3) \subseteq S(4)$, which proves the proposed second relationship ($v(F3) \geq v(F4)$).

We also see that $S(4) \subseteq S(5)$. The reason is that in $F5$ the allowable time for a setup crossover is restricted to the maximum of the active setups, whereas in $F4$ it is restricted to the minimum of the active setups, which is more restrictive.

To show that $v(F5) = v(F1)$ observe that there is an incentive to make the right-hand side of (4.15) as large as possible, in order to allow the maximum flexibility. The v_{it} variable does not appear in the objective function, and the values are constrained by inequalities (4.16), (4.17) and the domain restrictions. Therefore, there exists an optimal solution for $F1$ in which the right-hand side of (4.15) is $\sum_{i=1}^{n} v_{i,t-1}st_{it} = \max_{i \in I | y_{it}=1}\{st_{it}\}$. When we drop the integrality constraints on the v_{it} variables, the right-hand side of (4.15) will still have $\max_{i \in I | y_{it}=1}\{st_{it}\}$ as the maximum value. Therefore, by dropping the integrality constraints we will obtain the same objective function value as with the integrality constraints.

Finally, the formulations $F1$ and $F2$ are both valid for the same problem and hence provide the same optimal objective function value ($v(F1) = v(F2)$).■

Lemma 4.2 (related to symmetry breaking constraint on formulation $F1$) Given a feasible (or optimal) solution for $F1$, with a setup crossover for product i from period t to $t+1$ (i.e. $v_{it} = 1$ and $v_{jt} = 0\ \forall j \in I \setminus \{i\}$), we can construct an alternative feasible (or optimal) solution with the same objective function value if there exists in period $t+1$ an active setup for another product i' which has an equal or higher setup time (i.e. $st_{i',t+1} \geq st_{i,t+1}$ and $y_{i',t+1} = 1$). This solution can be constructed as follows:

$$v_{i't} = 1,\ v_{it} = 0 \text{ and all other variables (including } u_t) \text{ remaining the same.}$$

Proof: The proof is easily established by the following two reasons:
1) The new solution satisfies all the constraints;

2) We have the same objective function, because the values of the variables x_{itk} and y_{it} remain the same.■

Lemma 4.3 (related to symmetry breaking constraint on formulation $F2$) Given a feasible (or optimal) solution for $F2$, with a setup crossover for product i from period t to $t+1$ (i.e. $v_{i,t+1} = 1$, $v_{j,t+1} = 0\ \forall j \in I \setminus \{i\}$ and $z_{i,t+1} = 0$), we can construct an alternative feasible (or optimal) solution with the same objective function value if there exists in period $t+1$ an active setup for another product i' which has an equal or higher setup time (i.e. $st_{i',t+1} \geq st_{i,t+1}$ and $z_{i',t+1} = 1$). This solution can be constructed as follows:

$$v_{i',t+1} = 1,\ z_{i',t+1} = 0$$
$$v_{i,t+1} = 0,\ z_{i,t+1} = 1$$
$$l_{i't} \longleftarrow l_{it},\ l_{it} = 0$$
$$f_{i',t+1} = st_{i',t+1} - l_{i't},\ f_{i,t+1} = 0$$

Proof: The proof is established by the following two reasons:

1) The new solution remains feasible. Indeed:

(4.21) is satisfied for items i and i' since the x_{ikt} variables do not change;

The left-hand side of (4.22) for period t remains unchanged since $l_{i't}$ has taken the value of l_{it} and l_{it} has taken the value of zero, so that $\sum_{j=1}^{n} l_{jt}$ remains the same in the two solutions.

The left-hand side of (4.22) for period $t+1$ has the same value after the changes (see Table 4.1):

	Old solution	New solution
$\sum_{i=1}^{n} st_{i,t+1} z_{i,t+1}$	$st_{i',t+1} \times 1$	$st_{i,t+1} \times 1$
$\sum_{i=1}^{n} l_{i,t+1}$	unchanged	unchanged
$\sum_{i=1}^{n} f_{i,t+1}$	$f_{i,t+1} = st_{i,t+1} - l_{it}$	$f_{i',t+1} = st_{i',t+1} - l_{i't}$
TOTAL	$st_{i',t+1} + st_{i,t+1} - l_{it}$	$st_{i,t+1} + st_{i',t+1} - l_{i't}$

Table 4.1: Constraints (4.22) for period $t+1$ and items i and i'.

(4.23) is satisfied since the right-hand side for item i and i' is still equal to 1;

(4.24) is satisfied for items i and i' by construction;

(4.25) is satisfied since we still only have one item (in each period) for which we allow a setup crossover.

2) We have the same objective function by construction.■

4.4.2 Example

The following example shows the solutions for all formulations applied to the same instance. We adapted the example proposed in Belo-Filho et al. (2014) making some changes considering that in their case setup carryover is allowed.

We have to determine a production plan for four different items $i = \{A, B, C, D\}$ over a planning horizon composed of five non-uniform periods. Tables 4.2 and 4.3 contain the parameters, the demand and capacity values. Note that the parameters are time independent in Table 4.2.

	A	B	C	D
vt_i	0.1	0.1	0.1	0.1
hc_i	3	4	1	6
st_i	3	4	1	6
sc_i	3	4	1	6

Table 4.2: Parameters for the example.

d_{it}	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$i = A$	0	30	0	0	0
$i = B$	40	0	20	20	0
$i = C$	0	0	30	0	0
$i = D$	0	0	0	0	40
Cap_t	10	10	10	6	6

Table 4.3: Demand and capacity data.

All formulations were solved to optimality using this data set. Figure 4.1 and Table 4.4 illustrate the graphical solutions and the relevant non-zero variable values, respectively.

In Figure 4.1, white blocks represent production time that is consumed in that period, dark grey represents the setup time and light grey represents idle time. The values of the non-zero decision variables can be found in Table 4.4. For the classical formulation ($F0$) the value of the optimal solution is 688. This high value results mainly from the inventory for item B from period 1 to period 3 (20 units), for item C from period 2 to period 3 (30 units) and for item D from period 3 to period 5 (40 units). Note that in this example the inventory costs are very high. However, due to the lack of capacity it is impossible to have a setup for each item in a period with positive demand, and the optimal solution for $F0$ results in high inventory levels.

Although slightly different, the solutions found by the formulations $F1$, $F1'$, $F5$, $F2$ and $F2'$ have the same objective function value of 22. The only difference is that in formulation $F1$ and $F1'$ the setup for item A is split between periods 1 and 2 and for the formulations $F5$, $F2$ and $F2'$ the setup for this item is completely done in period 2. Observe that the solution obtained by $F2$, $F2'$ and $F5$ can directly be transformed into the solution obtained by $F1$ and $F1'$ by splitting the setup of product A over periods 1 and 2. The two solutions presented are both feasible for $F1$, $F1'$, $F2$, $F2'$ and $F5$. We see hence that there can be equivalent alternative optimal solutions. Note that there are no inventories in the solutions obtained by these formulations. Note also that although formulations $F2$ and $F2'$ present the same solution, the value of some variables are different (see Table 4.4). It occurs because for a product with a partial setup between periods $t-1$ and t ($v_{it} = 1$), it is still possible in some cases to do the complete setup in period t by choosing $l_{i,t-1} = 0$ or in period $t-1$ by choosing $f_{it} = 0$.

For the formulation $F3$ the maximum extra time that could be borrowed in each period was 1. Consequently, the optimal solution contains inventory for item B from period 1 to period 3 (20 units), for item C from period 2 to period 3 (30 units) and for item D from period 3 to period 5 (30 units). The optimal objective function value for this formulation is 574. When we limit the borrowed extra time to the minimum of the active setup times ($F4$), the formulation has more flexibility to find better solutions than $F3$. The optimal solution found by formulation $F4$ has inventory only for item C from period 2 to period 3 (30 units) and the objective function value of the optimal solution is 52. The results are in line with the relationships proposed in Lemma 1, and indicate that the inequalities can be strict.

4.5 Computational results

The formulations were modeled in AMPL using CPLEX 12.6 as solver. The tests were done on a personal computer Intel Core-I5, 2.27GHz with 6GB of RAM and the Windows operating system. The computational tests involve four experiments based on standard instances proposed in Trigeiro et al. (1989). In the first experiment, the formulations are tested for the well known F and G instances. In the second experiment, the formulations were tested on a large data set of 540 standard instances, in the third one we test some adapted instances with high values for the inventory costs and finally, in the fourth one we test some instances considering the possibility of backlog. We have limited the computational time in all experiments to 1800 seconds per instance. Note that in these instances the unit production costs are not considered.

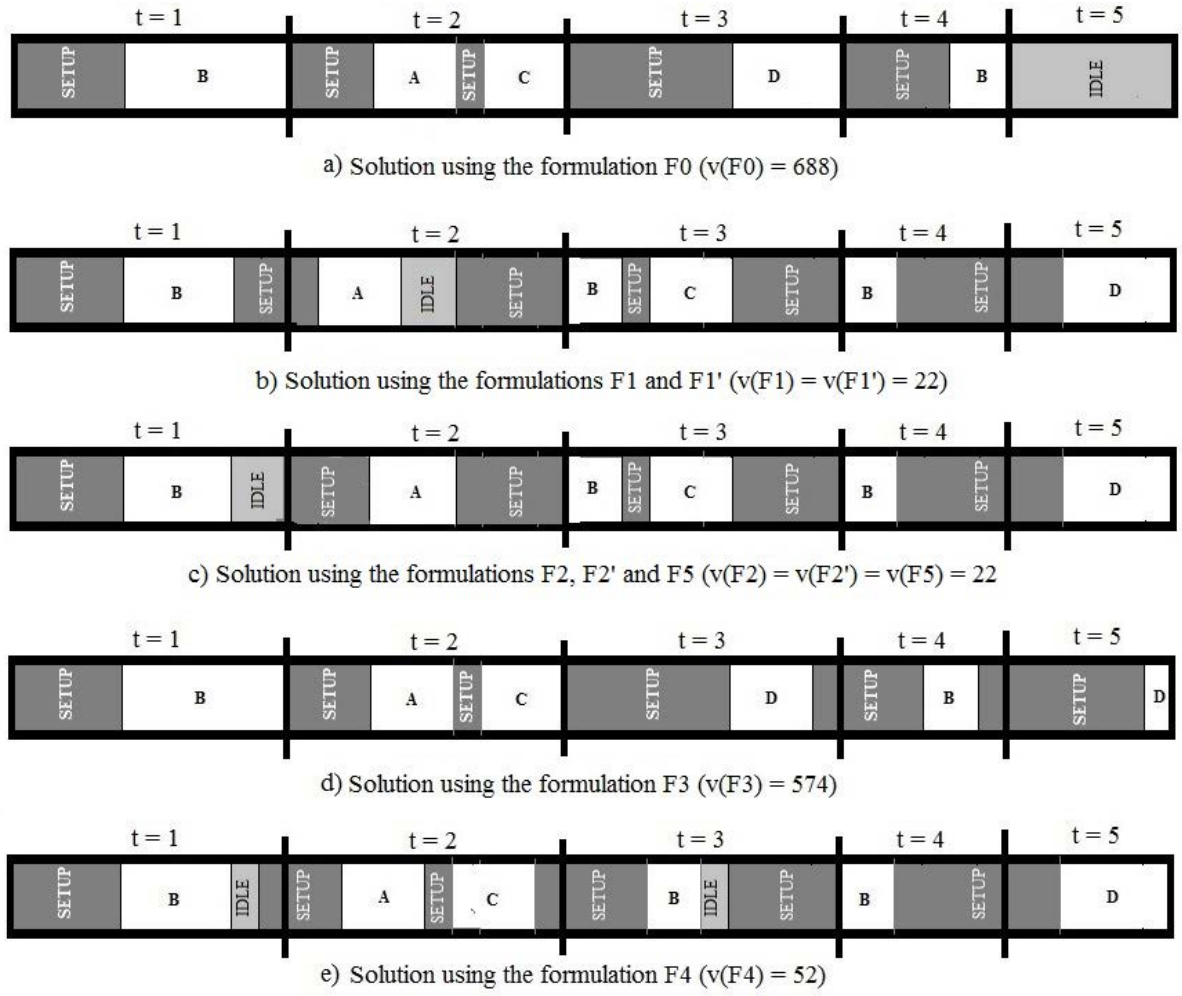


Figure 4.1: Graphical solution of the example for all formulations.

4.5.1 Results for experiment 1

The formulations were tested for a set of 145 instances proposed in Trigeiro et al. (1989). These are 70 instances from the F-set and 75 from the G-set. The F-set contains 70 instances with 6 items and 15 periods. The G-set consists of 50 instances with 6 items and 15 periods and 5 instances for each of the cases with 12 items and 15 periods, 24 items and 15 periods, 6 items and 30 periods, 12 items and 30 periods and 24 items and 30 periods.

In Table 4.5 we give the upper bounds (Columns UP) and the computational times in seconds (Columns $Time$) for all formulations. We set the upper bounds found by the classical formulation ($F0$) to 100% and calculate the others values relative to this. As expected, all formulations with setup crossover have found better solutions than the classical formulation ($F0$) and the differences are bigger for problems with 6 items. Comparing the computational times, we can see that all formulations except the formulations $F2$,

$F0$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B13}=1$	$x_{B44}=1$	$x_{C23}=1$	$x_{D35}=1$	
$y_{A2}=1$	$y_{B1}=1$	$y_{B4}=1$	$y_{C2}=1$	$y_{D3}=1$		
$F1$ and $F1'$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B33}=1$	$x_{B44}=1$	$x_{C33}=1$	$x_{D55}=1$	
$y_{A2}=1$	$y_{B1}=1$	$y_{B3}=1$	$y_{B4}=1$	$y_{C3}=1$	$y_{D5}=1$	
$u_1=2$	$u_2=4$	$u_3=4$	$u_4=4$			
$v_{A1}=1$	$v_{B2}=1$	$v_{B3}=1$	$v_{D4}=1$			
$F2$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B33}=1$	$x_{B44}=1$	$x_{C33}=1$	$x_{D55}=1$	
$z_{A2}=1$	$z_{B1}=1$	$z_{C3}=1$				
$v_{B3}=1$	$v_{B4}=1$	$v_{D5}=1$				
$l_{B2}=4$	$l_{B3}=4$	$l_{D4}=4$				
$f_{D5}=2$						
$F2'$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B33}=1$	$x_{B44}=1$	$x_{C33}=1$	$x_{D55}=1$	
$z_{C3}=1$						
$v_{A2}=1$	$v_{B1}=1$	$v_{B3}=1$	$v_{B4}=1$	$v_{D5}=1$		
$l_{B2}=4$	$l_{B3}=4$	$l_{D4}=4$				
$f_{A2}=3$	$f_{B1}=4$	$f_{D5}=2$				
$F3$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B13}=1$	$x_{B44}=1$	$x_{C23}=1$	$x_{D35}=0.75$	$x_{D55}=0.25$
$y_{A2}=1$	$y_{B1}=1$	$y_{B4}=1$	$y_{C2}=1$	$y_{D3}=1$	$y_{D4}=1$	
$u_3=1$	$u_4=1$					
$F4$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B33}=1$	$x_{B44}=1$	$x_{C23}=1$	$x_{D55}=1$	
$y_{A2}=1$	$y_{B1}=1$	$y_{B3}=1$	$y_{B4}=1$	$y_{C2}=1$	$y_{D5}=1$	
$u_1=1$	$u_2=1$	$u_3=4$	$u_4=4$			
$F5$ variables						
$x_{A22}=1$	$x_{B11}=1$	$x_{B33}=1$	$x_{B44}=1$	$x_{C33}=1$	$x_{D55}=1$	
$z_{A2}=1$	$y_{B1}=1$	$y_{B3}=1$	$y_{B4}=1$	$y_{C3}=1$	$y_{D5}=1$	
$u_2=4$	$u_3=4$	$u_4=4$				
$v_{A1}=1$	$v_{B2}=1$	$v_{B3}=1$	$v_{D4}=0.66$			

Table 4.4: Variables values of example for all formulations.

$F2'$ and $F4$ for some instances are faster than the classical formulation.

We observe that $F1$ is much faster than $F0$, which is surprising since it contains more binary variables. Omitting the binary condition on the v_{it} variables as done in formulation $F5$ does not result in a significant change in the CPU time. It only provides in average a very small decrease compared to $F1$. We observe that $F3$ and $F4$, i.e. the two restricted models without binary variables to indicate the crossover take significantly more time to be solved compared to formulations $F1$ and $F5$.

Note that there is no benefit in adding symmetry breaking constraints to formulation $F1$ considering that the CPU times of the formulation $F1'$ are bigger than $F1$. However, for formulation $F2$ the symmetry breaking constraints are very efficient and the difference of CPU times between the formulations $F2$ and $F2'$ are very significant.

Note also that these instances are quite easy, considering that CPLEX has solved relatively fast almost all instances for all formulations except formulation $F2$. Moreover,

with formulations $F1$, $F1'$, $F2'$, $F4$ and $F5$ the solver has proven the optimality for all instances within the time limit. Using formulations $F0$, $F2$ and $F3$ the solver has proven the optimality for 98.6%, 88.9% and 99.3%, respectively.

	F		G6-15		G12-15		G24-15		G6-30		G12-30		G24-30	
Model	UP	Time	UP	Time	UP	Time	UP	Time	UP	Time	UP	Time	UP	Time
F0	100	2.3	100	21.7	100	5.8	100	10.2	100	364.3	100	642.6	100	472.9
F1	99.36	1.1	99.03	3.4	99.86	3.3	99.85	7.3	99.38	17.3	99.72	82.3	99.97	89.7
F1'	99.36	1.3	99.03	3.5	99.86	4.3	99.85	11.6	99.38	24.7	99.72	112.0	99.97	131.4
F2	99.36	87.0	99.03	102.9	99.86	401.1	99.85	569.1	99.38	936.4	99.72	1444.4	99.97	1332.5
F2'	99.36	7.7	99.03	21.5	99.86	16.0	99.85	17.7	99.38	136.0	99.72	211.3	99.97	228.7
F3	99.72	1.6	99.65	8.6	99.95	3.9	99.93	8.3	99.73	81.1	99.92	411.3	99.99	446.6
F4	99.67	1.3	99.54	10.7	99.95	10.0	99.90	15.7	99.73	237.5	99.90	365.6	99.99	390.0
F5	99.36	0.9	99.03	3.1	99.86	3.6	99.85	7.5	99.38	16.5	99.72	82.4	99.97	83.3

Table 4.5: Average general results for F and G data sets.

4.5.2 Results for experiment 2

In this experiment, the formulations were tested on a total of 540 instances with 20 periods, such that five characteristics are analyzed: number of items (10, 20 and 30), demand variability (medium $[0, 125]$ and high $[0, 200]$), setup cost (low $[25, 75]$, medium $[100, 300]$ and high $[400, 1200]$), setup time (low $[5, 17]$ and high $[21, 65]$) and capacity utilization (low [75%], medium [85%] and high [95%]). The numbers in the brackets indicate a uniform distribution between the two numbers. For more details on the data set, we refer to Trigeiro et al. (1989).

Tables 4.6 to 4.10 show the overall performance of the formulations. We report the relative upper bounds (UP), computational times in seconds ($Time$) and percentage of instances solved to optimality within the limit of 1800 seconds (OS). Since the symmetry breaking constraints in $F1'$ were not able to improve the results obtained by $F1$, the results of $F1'$ are omitted.

The overall analysis of Tables 4.6 to 4.10 confirm the tendencies observed in Table 4.5. $F0$ is slower than $F1$, $F3$, $F4$ (except for 20 items) and $F5$. $F2$ is overall the slowest formulation and the performance of $F2'$ is in fact significantly better than $F2$. We see that $F1$ and $F5$ generally provide a similar performance (both in terms of CPU times and the percentage of optimal solutions found). $F3$ and $F4$ provide a significantly worse performance compared to $F1$ and $F5$. We also observe that the cost decrease obtained by introducing the possibility of a setup crossover is very small in these instances and that the relevance of including a setup crossover is bigger for problems with few items. The average cost decrease is 0.59% for 10 items, 0.23% for 20 items and 0.17% for 30 items. The average cost decrease over all 540 instances is 0.33%.

Table 4.6 shows that although the formulations $F1$, $F2$, $F2'$ and $F5$ have the same optimal solutions, $F5$ found slightly better solutions for problems with 20 and 30 items.

Considering only instances for which CPLEX proved optimality (column Aver. OS) we clearly see the big improvement obtained by including the symmetry breaking constraints in the formulation $F2$. The CPU times for formulations $F2$ and $F2'$ are 96.1 and 9.5, respectively.

	10 items			20 items			30 items			Aver. OS	
Model	UP	Time	OS	UP	Time	OS	UP	Time	OS	UP	Time
F0	100	589.8	70.6	100	684.9	63.9	100	707.1	62.8	100	10.7
F1	99.41	462.0	80.6	99.78	625.2	68.9	99.86	613.0	68.9	99.88	4.7
F2	99.43	813.7	57.2	99.79	859.5	53.9	99.89	865.3	55.0	99.88	96.1
F2'	99.43	628.0	68.8	99.79	685.0	64.5	99.89	672.8	65.5	99.88	9.5
F3	99.75	530.0	75.0	99.90	668.9	65.0	99.94	668.6	66.1	99.95	5.9
F4	99.73	539.1	73.4	99.90	688.3	63.9	99.94	666.8	65.6	99.94	6.3
F5	99.41	475.9	78.3	99.77	604.5	70.0	99.83	609.9	68.3	99.88	4.6

Table 4.6: General average results aggregated per number of items.

Table 4.7 shows that the capacity utilization is an important factor for the quality of solutions using setup crossover and the difficulty of the problems. For problems with loose capacity, the inclusion of a setup crossover is not so important. The problems are quite easy considering that the computational times are low and the percentage of solutions solved to optimality (OS) is very high. When the capacity is tight, the importance of including setup crossover and the difficulty of the problems increase. Note that the computational times are very high and the percentage of instances that CPLEX solved to optimality is very low for these instances.

	Loose Capacity			Normal Capacity			Tight Capacity		
Model	UP	Time	OS	UP	Time	OS	UP	Time	OS
F0	100	29.3	98.9	100	404.8	84.4	100	1547.8	15.0
F1	99.95	17.2	99.4	99.79	280.7	91.1	99.30	1402.3	27.2
F2	99.95	126.2	96.1	99.79	777.3	61.7	99.37	1634.9	9.45
F2'	99.95	27.1	99.4	99.79	435.7	81.6	99.37	1520.4	17.8
F3	99.98	25.7	98.9	99.91	347.8	87.2	99.70	1493.9	20.0
F4	99.98	25.7	98.9	99.90	354.9	85.6	99.69	1513.7	18.3
F5	99.95	16.0	99.4	99.79	276.9	90.6	99.28	1397.4	26.7

Table 4.7: General average results aggregated per capacity.

Table 4.8 presents the results aggregated per setup cost level. The benefits of including a setup crossover decreases when the value of setup cost increases. It occurs because if the setup cost is high, the formulations try to reduce the numbers of setups and keep more items in inventory whereas one of the main gains of setup crossover is exactly the flexibility to produce as close as possible to the demand period avoiding big quantities of inventory. Note also that computational times increase and the OS decrease significantly when the value of setup cost increases.

Tables 4.9 and 4.10 show the results taking into account the setup time and demand variability, respectively. In these tables we can see that with increasing setup times

	Low setup cost			Normal setup cost			High setup cost		
Model	UP	Time	OS	UP	Time	OS	UP	Time	OS
F0	100	477.8	74.4	100	623.3	67.2	100	880.8	57.2
F1	99.55	384.1	81.7	99.75	521.2	73.3	99.74	764.9	62.8
F2	99.56	550.4	70.0	99.76	772.8	60.0	99.79	1215.2	36.1
F2'	99.56	470.6	75.0	99.76	572.4	69.4	99.79	858.5	58.8
F3	99.82	440.9	78.3	99.89	594.6	68.3	99.88	832.1	60.6
F4	99.82	455.6	76.7	99.88	600.6	68.3	99.87	828.1	59.4
F5	99.52	378.6	81.1	99.74	545.3	73.3	99.74	766.4	62.8

Table 4.8: General average results aggregated per setup cost level.

and demand variability, the computational times decrease, OS increases and the relative upper bounds decrease, indicating a larger benefit provided by the crossover. Note that for problems with low setup times and low demand variability the proposed formulation $F5$ found again slightly better solutions than the formulations from the literature $F1$ and $F2$.

	Low setup time			High setup time		
Model	UP	Time	OS	UP	Time	OS
F0	100	715.6	64.8	100	605.7	69.6
F1	99.78	669.9	68.1	99.58	463.5	77.8
F2	99.80	931.3	51.4	99.60	761.0	59.6
F2'	99.80	764.7	65.5	99.60	561.8	72.2
F3	99.91	699.1	63.7	99.81	545.9	73.7
F4	99.91	710.9	62.6	99.80	551.9	72.9
F5	99.76	670.1	66.3	99.58	456.8	78.1

Table 4.9: General average results aggregated per setup time.

	Medium demand variability			High demand variability		
Model	UP	Time	OS	UP	Time	OS
F0	100	728.2	62.6	100	593.1	70.0
F1	99.72	645.9	67.8	99.64	487.6	77.0
F2	99.75	844.4	55.2	99.65	847.9	55.6
F2'	99.75	735.7	63.3	99.65	585.5	71.8
F3	99.88	702.1	63.7	99.85	542.9	73.7
F4	99.88	704.2	63.3	99.83	558.6	72.2
F5	99.70	644.7	67.0	99.64	482.1	77.4

Table 4.10: General average results aggregated per demand variability.

Aiming to do a further analysis of the effect of introducing a setup crossover, Table 4.11 shows the behavior of the solutions for 10 items for the formulations $F0$ and $F1$: the percentage of setup and holding cost (columns $SC(\%)$ and $HC(\%)$) in the objective function value and the number of setups and total inventory (columns setup and inv.). We observe that overall the number of setups is very similar for both formulations and the main difference is the level of inventory. We obtain a decrease in total inventory of approximately 1%, but the total inventory holding costs constitute only 30% of the

total cost. So the overall cost decrease is relatively small. It explains the small decrease obtained by introducing the possibility of a setup crossover in these instances given that the value of the inventory costs are very low. Even though globally the total number of setups does not significantly change when we introduce a setup crossover, we see that the setup cost and the capacity tightness have an impact. Tight capacity levels and low setup cost generally lead to a slight increase in the total number of setups when allowing a setup crossover. The level of the setup times and the demand variability do not have a large impact. The overall analysis of these instances indicates that the benefits of a setup crossover come mainly from the decreased inventory level which results from a better matching of demand and supply through the increased flexibility. This might require, however, a slight increase in the number of setups as well.

		Model F0				Model F1			
		SC(%)	HC(%)	setup	inv.	SC(%)	HC(%)	setup	inv.
Capacity	Loose	74.75	25.24	101.7	10931	74.78	25.21	101.6	10880
	Normal	72.87	27.13	101.9	11133	73.04	26.95	101.9	11120
	Tight	62.81	37.19	95.3	12452	63.87	36.17	95.7	12158
S. cost	Low	81.63	18.37	151	2051	82.56	17.44	151.6	1932
	Normal	65.33	34.67	90.4	9232	65.64	34.36	90.4	9118
	High	63.46	36.54	57.6	23234	63.47	36.53	57.2	23108
S. time	Low	69.05	30.95	98.8	11516	69.36	30.64	98.8	11467
	High	71.23	28.77	100.6	11435	71.75	28.25	100.7	11305
Demand	Medium	72.07	27.93	108.3	11516	72.30	27.70	108.2	11485
	High	68.21	31.79	91.1	11495	68.81	31.19	91.3	11286
Average		**70.14**	**29.86**	**99.6**	**11505**	**70.56**	**29.44**	**99.7**	**11386**

Table 4.11: Detailed results for 10 items.

It is important to note that although the cost decrease obtained by introducing the possibility of a setup crossover is on average small in these instances (i.e. 0.33%), for some cases this decrease is more relevant. Tables 4.12 and 4.13 contain the results for all 108 different combinations (organized according to increasing upper bound) considering the five characteristics discussed in this experiment. Note that for each of the 108 combinations, 5 instances were tested. We observe that the cost decrease obtained by introducing the possibility of a setup crossover is the biggest for configurations with 10 items, tight capacity, low setup cost, high setup time and medium and high demand variability, where we obtained a 2.77% and 2.62% cost decrease by including a setup crossover. On the other hand, for many configurations with loose capacity we did not obtain any improvement. This is in line with the aggregated analysis presented in Tables 4.6 to 4.10.

4.5.3 Results for experiment 3

In this experiment the formulations were tested on a set of 180 instances. These are the same as the instances with 10 items of experiment 2 with an altered high value for the inventory costs. To generate the instances with high inventory costs we multiply the inventory costs by 10 and 100.

Characteristic of the problem					Formulation F5		
Items	Cap.	S. Cost	S. Time	Demand	UB	Time	OS
10	T	L	H	M	97.23	864.8	60
10	T	L	H	H	97.38	2.7	100
30	T	L	L	M	98.26	1800	0
10	T	H	H	H	98.34	1800	0
10	T	L	L	M	98.53	1561.7	20
10	T	M	H	H	98.59	661.4	80
10	T	H	L	M	98.65	1800	0
10	T	L	L	H	98.71	618.9	80
10	M	L	H	H	98.71	0.3	100
10	T	M	H	M	98.86	1714.5	20
10	T	H	H	M	98.88	1440.0	20
20	T	L	H	H	99.02	72.8	100
20	T	L	H	M	99.12	1226.4	60
20	T	M	H	H	99.29	1211.2	60
20	T	H	H	M	99.30	1800	0
10	T	M	L	H	98.32	1800	0
30	T	H	H	M	99.39	1800	0
10	M	H	H	H	99.40	223.6	100
20	T	L	L	H	99.40	1445.5	20
20	T	H	H	H	99.41	1800	0
10	T	M	L	H	99.42	112.0	60
30	T	L	H	M	99.44	1494.5	20
20	T	M	H	M	99.45	1800	0
10	T	H	L	H	99.46	1690.9	20
20	T	L	L	M	99.47	1800	0
10	M	L	L	M	99.48	0.6	100
30	T	L	L	H	99.53	1800	0
10	M	H	H	M	99.59	332.6	100
30	T	L	H	H	99.60	14.5	100
10	M	M	H	H	99.61	1.4	100
30	T	M	H	M	99.62	1800	0
30	T	H	H	H	99.63	1800	0
10	M	L	H	M	99.66	0.2	100
30	T	M	H	H	99.70	842.4	60
20	T	M	L	H	99.72	1800	0
20	T	H	L	M	99.73	1800	0
20	T	M	L	H	99.73	1800	0
10	M	M	H	M	99.74	2.26	100
10	M	H	L	H	99.74	48.7	100
20	M	L	H	H	99.74	0.7	100
10	L	H	H	H	99.76	2.6	100
30	T	H	L	H	99.77	1800	0
10	T	H	L	M	99.78	1800	0
10	M	M	L	H	99.79	16.3	100
10	L	H	H	M	99.82	6.1	100
30	T	M	L	H	99.82	1800	0
20	M	H	H	H	99.83	1.6	100
20	M	L	L	H	99.84	2.4	100
20	M	H	H	M	99.84	847.6	60
20	T	H	L	H	99.84	1800	0
30	T	M	L	M	98.84	1800	0
10	L	M	L	H	99.85	0.6	100
10	L	M	H	H	99.85	0.6	100
10	L	L	L	H	99.86	0.2	100
20	M	H	H	H	99.86	85.6	100
10	L	L	H	H	99.88	0.1	100
10	M	M	L	M	99.89	5.7	100
10	L	M	H	M	99.89	0.6	100
30	M	L	H	H	99.89	0.8	100
30	M	L	L	H	99.91	1.6	100
30	M	H	H	H	99.91	368.6	100
10	M	H	L	M	99.92	836.3	80
10	L	H	L	H	99.92	8.7	100
20	M	M	H	M	99.92	4.0	100
30	M	M	H	H	99.92	14.5	100
20	M	L	H	M	99.93	0.3	100
20	M	M	L	H	99.93	21.2	100
20	L	H	H	H	99.94	6.4	100
20	L	M	H	H	99.94	0.6	100

Table 4.12: General average results aggregated.

Characteristic of the problem					Formulation F5		
Items	Cap.	S. Cost	S. Time	Demand	UB	Time	OS
10	M	L	L	M	99.95	0.2	100
20	M	H	L	M	99.95	1247.3	40
20	M	H	L	H	99.95	584.5	80
30	M	H	H	M	99.95	854.7	60
20	L	L	H	H	99.96	0.3	100
20	L	H	H	M	99.96	17.4	100
30	M	M	H	M	99.96	8.2	100
30	M	H	L	H	99.96	270.3	100
30	L	H	H	H	99.96	5.2	100
10	L	M	L	M	99.97	0.7	100
10	L	H	L	M	99.97	15.5	100
20	M	M	L	M	99.97	11.5	100
20	L	L	L	H	99.97	0.5	100
20	L	H	L	H	99.97	5.4	100
30	M	H	L	M	99.97	1369.1	40
30	L	H	H	M	99.97	9.2	100
20	L	M	L	H	99.98	0.5	100
20	L	M	L	M	99.98	1.1	100
20	L	M	H	H	99.98	0.7	100
20	L	H	L	M	99.98	16.8	100
30	M	L	H	M	99.98	0.4	100
30	M	M	L	M	99.98	33.4	100
30	M	M	L	H	99.98	20.1	100
30	L	M	H	H	99.99	7.3	100
20	M	L	L	M	99.99	0.4	100
30	L	M	L	M	99.99	0.6	100
30	L	M	H	M	99.99	1.0	100
30	L	H	L	M	99.99	15.7	100
30	L	L	L	H	99.99	0.3	100
30	L	M	L	H	99.99	1.1	100
30	L	H	L	H	99.99	10.4	100
10	L	L	L	M	100	0.1	100
10	L	L	H	M	100	0.1	100
20	L	L	L	M	100	0.2	100
20	L	L	H	M	100	0.2	100
30	L	L	L	M	100	0.3	100
30	L	L	H	M	100	0.3	100
30	L	L	H	H	100	0.3	100
30	M	L	L	M	100	0.3	100

Table 4.13: General average results aggregated.

Tables 4.14 and 4.15 show the benefits of considering setup crossover for problems where the inventory costs are significant. The global analysis confirms some of the conclusions of the previous experiments. $F1$ and $F5$ have a similar performance. $F3$ and $F4$ are slower compared to $F1$ and $F5$, but faster than $F0$. $F2$ is again the slowest formulation. However, in contrast to the results of the previous experiments, we do see a significant decrease in the total costs when setup crossover is allowed, which is on average almost 3% (for the inventory cost $\times$ 10) and 8% (for the inventory cost $\times$ 100).

Table 4.14 contains the results for the instances with the inventory costs multiplied by 10. We observe that the benefits of a setup crossover are the highest in a setting with tight capacity (4.2% cost decrease), high setup times (4.4% cost decrease) and low setup cost (4.5% cost decrease). The analysis further reveals that the approximate formulations $F3$ and $F4$ are not able to capture all of the benefits with an average cost decrease of 1.3%.

Table 4.15 contains the results for instances with inventory costs multiplied by 100. We observe that for these instances, the effect of allowing a setup crossover is the highest

		Model F0		Model F1		Model F2		Model F2'		Model F3		Model F4		Model F5	
		UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)
Capacity	Loose	100	81	99.00	2	99.00	185	99.00	66	99.54	46	99.53	43	99.00	2
	Normal	100	499	96.40	342	96.41	926	96.41	579	98.42	363	98.37	393	96.40	331
	Tight	100	1313	95.79	1150	95.82	1683	95.79	1428	98.25	1240	98.12	1292	95.78	1138
S. cost	Low	100	292	95.49	225	95.49	604	95.49	436	98.19	265	98.16	268	95.49	225
	Normal	100	727	97.05	485	97.06	1036	97.04	758	98.58	574	98.54	619	97.04	473
	High	100	875	98.64	785	98.67	1154	98.67	880	99.44	811	99.33	842	98.64	775
S. time	Low	100	496	98.53	468	98.55	842	98.54	553	99.46	490	99.42	524	98.53	461
	High	100	766	95.60	528	95.61	1021	95.59	719	98.01	610	97.93	628	95.58	521
Demand	Medium	100	811	97.54	667	97.55	944	97.54	872	98.88	719	98.83	725	97.53	666
	High	100	451	96.59	330	96.60	920	96.59	511	98.59	381	98.52	427	96.59	316
Average		**100**	**631**	**97.06**	**498**	**97.08**	**932**	**97.07**	**691**	**98.74**	**550**	**98.67**	**576**	**97.06**	**491**

Table 4.14: Average general results with inventory costs multiplied by 10.

for the instances with normal capacity, low setup cost and high setup time where the total cost decreases more than 10% on average. Note also that even in a setting with loose capacity the total cost decrease is 6.3%.

		Model F0		Model F1		Model F2		Model F2'		Model F3		Model F4		Model F5	
		UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)	UP	T(s)
Capacity	Loose	100	125	93.71	34	93.72	216	93.71	129	97.22	72	97.21	80	93.71	34
	Normal	100	514	89.62	332	89.64	931	89.62	556	95.20	395	95.13	408	89.62	309
	Tight	100	1168	94.03	1016	94.06	1683	99.04	1365	97.49	1068	97.39	1274	94.02	992
S. cost	Low	100	290	89.82	222	89.83	604	89.82	384	95.65	246	95.61	268	89.82	214
	Normal	100	593	92.04	399	92.06	1037	92.04	740	96.12	454	96.08	633	92.04	382
	High	100	924	95.50	760	95.54	1189	95.50	927	98.14	834	98.04	860	95.49	740
S. time	Low	100	399	95.88	389	95.90	835	95.89	595	98.44	387	98.42	500	95.88	382
	High	100	805	89.02	532	89.04	1052	89.02	772	94.83	636	94.73	675	89.02	508
Demand	Medium	100	787	93.31	658	93.35	970	93.32	892	96.67	691	96.63	754	93.31	635
	High	100	417	91.59	263	91.60	917	91.59	475	96.60	332	96.52	421	91.59	255
Average		**100**	**602**	**92.45**	**460**	**92.47**	**943**	**92.45**	**683**	**96.63**	**512**	**96.57**	**587**	**92.45**	**445**

Table 4.15: Average general results with inventory costs multiplied by 100.

Table 4.16 and 4.17 show the behavior of the solutions for the results with inventory costs multiplied by 10 and 100. We observe that, contrary to the results of Table 4.11, the percentage of inventory cost in the objective function value is very high especially for instances in which the inventory costs are multiplied by 100 (62.74% for formulation $F0$ and 59.89% for formulation $F1$). It explains the more significant decrease obtained by introducing the possibility of a setup crossover in these instances given that the value of the inventory costs are very high. We observe that for the instances in which the inventory costs are multiplied by 10 (Table 4.16) the total inventory goes down by approximately 5%, and the total setups only increase by 0.5%. Regarding the instances in which the inventory costs are multiplied by 100 (Table 4.17) the total inventory goes down by approximately 6% and the total setups only increase by 0.5%.

4.5.4 Results for experiment 4

In this experiment the formulations were adapted to allow backlog and were tested on a set of 60 instances. These are the same as the instances with 10 items and tight capacity of experiment 2 with an altered (reduced) value for the capacity in order to generate some

		Model F0				Model F1			
		SC(%)	HC(%)	setup	inv.	SC(%)	HC(%)	setup	inv.
Capacity	Loose	88.33	11.66	158.8	1312	89.64	10.35	159.4	1200
	Normal	64.39	35.60	145.4	3085	66.88	33.11	146.3	2868
	Tight	25.25	74.74	113.3	9260	26.51	73.48	111.8	8881
S. cost	Low	62.99	37.00	163.7	1603	65.26	34.73	164.4	1461
	Normal	57.04	42.95	139.0	4152	58.89	41.10	139.8	3901
	High	57.95	42.04	112.8	7902	58.88	41.11	113.1	7588
S. time	Low	61.42	38.57	141.2	4144	62.16	37.83	141.5	4031
	High	57.23	42.76	135.8	4961	59.86	40.13	136.8	4602
Demand	Medium	64.62	35.37	147.3	4198	66.01	33.98	148.0	3972
	High	54.03	45.96	129.7	4907	56.01	43.98	130.3	4661
Average		**59.33**	**40.66**	**138.5**	**4552**	**61.01**	**38.98**	**139.1**	**4316**

Table 4.16: General detailed results with inventory costs multiplied by 10.

		Model F0				Model F1			
		SC(%)	HC(%)	setup	inv.	SC(%)	HC(%)	setup	inv.
Capacity	Loose	74.52	25.48	170.4	712	78.69	21.31	170.9	597
	Normal	31.18	68.82	152	2810	35.21	64.79	153	2562
	Tight	6.06	93.94	114.9	9316	6.43	93.57	115.4	8905
S. cost	Low	45.49	54.51	164.9	1602	49.15	50.85	165.8	1458
	Normal	32.59	67.41	143.5	4144	35.86	64.14	144.0	3877
	High	33.69	66.31	128.8	7093	35.32	64.68	129.5	6728
S. time	Low	43.84	56.16	150.9	3768	45.76	54.24	151.2	3641
	High	30.67	69.33	140.7	4791	34.46	65.54	141.7	4022
Demand	Medium	47.47	52.53	152.6	3893	50.54	49.46	153.0	3659
	High	27.04	72.96	138.9	4666	29.68	70.32	139.8	4384
Average		**37.26**	**62.74**	**145.8**	**4280**	**40.11**	**59.89**	**146.5**	**4021**

Table 4.17: General detailed results with inventory costs multiplied by 100.

backlog. To generate these instances with very tight capacity we reduce the capacity by 5% and 10%. We set the backlog costs for each item equal to 100× inventory holding cost.

Tables 4.18 to 4.21 present the overall performance of the formulations for problems that consider the possibility of backlog (based on instances in which the two formulations found feasible solutions). We report all factors that have been analyzed and added the percentage of backlog cost (columns $B(\%)$) in the total objective function value, the percentage of feasible solutions (columns FS) and the number of total backlog (columns Back.).

The overall analysis of Tables 4.18 to 4.21 show that for problems that allow the possibility of backlog there is a significant decrease in the total costs when a setup crossover is allowed, which is on average 2.3% and 4% for instances for which the capacity is reduced by 5% and 10%, respectively. We also observe, especially for instances for which the capacity is reduced by 10%, an increase in the number of feasible solutions (4%).

Tables 4.18 and 4.19 present the results for instances for which the capacity is reduced by 5%. We observe that for these instances when the setup cost and time is high, the importance of including a setup crossover increase. Note also that overall the number of setups is very similar again for the case with and without setup crossover. We obtain a decrease in total inventory and backlog of approximately 2.7% and 4%, respectively. Finally, the percentage of backlog in the total objective function value is relatively small

(on average only 11%) and there is no backlog for instances with low setup cost.

		Model F0			Model F1		
		UP	T(s)	FS	UP	T(s)	FS
S. cost	Low	100	1401	100	97.86	1101	100
	Normal	100	1800	100	98.55	1666	100
	High	100	1715	80	96.58	1731	85
S. time	Low	100	1742	86	99.08	1758	90
	High	100	1539	100	96.57	1245	100
Demand	Low	100	1800	93	97.86	1617	97
	High	100	1467	93	97.62	1350	93
Average		**100**	**1633**	**93**	**97.73**	**1483**	**95**

Table 4.18: General results with backlog and capacity reduced by 5%.

		Model F0						Model F1					
		SC(%)	HC(%)	B(%)	setup	Inv.	Back.	SC(%)	HC(%)	B(%)	setup	Inv.	Back.
S. cost	L	45.52	54.48	0	114.8	7676	0	46.97	53.03	0	115.5	7397	0
	N	47.59	50.34	2.05	77.5	15345	10	48.84	49.17	1.97	78.2	14766	10
	H	43.25	21.02	35.72	60.3	21167	793	44.71	21.48	33.80	60.3	20921	759
S. time	L	40.98	51.43	7.57	83.5	14911	252	41.55	50.92	7.51	83.7	14639	252
	H	49.63	36.51	13.85	88.0	13714	211	51.71	35.46	12.82	88.8	13246	192
Demand	M	44.44	41.86	13.69	90.3	14143	341	45.73	41.08	13.18	90.9	13739	335
	H	46.79	45.18	8.18	81.6	14397	120	48.25	44.19	7.54	81.9	14047	106
Average		**45.61**	**43.44**	**10.93**	**85.9**	**14270**	**230**	**46.99**	**42.64**	**10.36**	**86.4**	**13893**	**221**

Table 4.19: General detailed results with backlog capacity reduced by 5%.

Tables 4.20 and 4.21 show the results for instances in which the capacity is reduced by 10%. For these instances the percentage of backlog in the total objective function value is more relevant (approximately 30%) and we obtain a decrease in total backlog of 10.2%. We observe that for instances with high setup cost, the percentage of backlog in the objective function value is 93%. Moreover, we obtained a decrease in the total costs of 8.8% for these instances.

		Model F0			Model F1		
		UP	T(s)	FS	UP	T(s)	FS
S. cost	Low	100	1731	100	97.63	1411	100
	Normal	100	1800	55	95.21	1688	55
	High	100	1800	25	91.23	1800	35
S. time	Low	100	1800	37	96.57	1800	37
	High	100	1745	83	95.75	1440	90
Demand	Medium	100	1800	66	94.92	1800	70
	High	100	1714	53	97.36	1237	57
Average		**100**	**1762**	**59**	**96.00**	**1550**	**63**

Table 4.20: General results with backlog capacity reduced by 10%.

		Model F0						Model F1					
		SC(%)	HC(%)	B(%)	setup	Inv.	Back.	SC(%)	HC(%)	B(%)	setup	Inv.	Back.
S. cost	L	27.68	48.80	23.50	91.2	12399	781	29.01	47.58	23.39	92.4	12083	775
	N	35.58	50.34	14.06	65.3	19995	1152	36.82	49.38	13.79	65.3	19021	959
	H	3.08	3.86	93.04	33.2	28080	7925	3.38	4.03	92.57	33.4	26682	6977
S. time	L	9.10	39.31	51.58	66.9	19588	2525	9.25	39.43	51.30	66.8	19164	2324
	H	34.41	44.67	20.91	78.9	15714	1606	36.02	43.25	20.72	80.0	14940	1415
Demand	M	22.78	39.22	37.99	73.5	18012	2625	23.68	38.61	37.69	73.9	17205	2299
	H	31.55	47.80	20.64	77.4	15505	964	33.03	46.43	20.52	78.6	15012	935
Average		**26.68**	**43.03**	**30.28**	**75.2**	**16898**	**1886**	**27.84**	**42.08**	**30.06**	**76.0**	**16231**	**1693**

Table 4.21: General detailed results with backlog capacity reduced by 10%.

4.6 Conclusion

In this chapter, the lot sizing problem with capacity constraints and setup crossover was studied. A reformulation of the problem using the simple plant location model was used. Three new formulations avoiding the necessity to define new extra binary variables to model the setup crossover and two adding new symmetry breaking constraints were proposed. Using CPLEX 12.6 these formulations were compared with two different formulations proposed in literature to model the setup crossover using extra binary variables and with the classical assumption where a setup crossover is not allowed. The results show that the proposed formulations are efficient, specially the formulation $F5$ which is slightly better than the formulation $F1$ proposed in the literature. Comparing the benefits obtained allowing a setup crossover with the classical assumption, we conclude that it depends on the characteristic of the problem, but especially for problems with high inventory cost it can be very significant. Indeed, when the inventory costs were multiplied by 10 and 100 the cost decrease obtained by introducing the possibility of a setup crossover are on average 3% and 7.5%. Finally, we also conclude that the benefits obtained allowing a setup crossover are significant for problems which allow the possibility of backlog. We obtained a decrease in the total costs by 2.3% and 4% when the capacity was decreased by 5% and 10%, respectively. Moreover, the formulation with a setup crossover found more feasible solutions.

Chapter 5

Conclusions and future works

In this thesis we have studied two extensions of the lot sizing problem. The first one is the lot sizing problem with unrelated parallel machines, in which items can be produced in any of the machines and to start producing an item incurs a time and cost for set up the machine used. The second extension is the lot sizing problem with setup crossover in which the first setup operation of each planning period can already start in the previous period. Note that the goal of the two extensions discussed is to find a production plan with minimum cost that is able to meet the demand of items without exceeding the capacity of the machine(s).

For the lot sizing problem with unrelated parallel machines a reformulation is proposed using the ideas of redefinition of variables proposed by Eppen and Martin (1987), which allows to treat the problem as a shortest path problem. Three solution methods to find lower bounds that use Dantzig-Wolfe decomposition and Lagrangian relaxation were extended from the literature. In all methods, the Lagrangian relaxation is applied to the flow constraints and the Dantzig-Wolfe decomposition is applied using as linking constraints the inventory balance constraints. The problem decomposes in independent subproblems by periods and machines instead of classical decomposition by items. As we relax the inventory balance constraints, the solutions obtained by the methods respect the capacity of the machines but may be infeasible regarding the constraints to meet the demand. Therefore, to find feasible solutions we have proposed two Lagrangian heuristic based on transfer of production which aim the feasibility of such constraints.

These methods were tested for different types of instances and the results were compared with results from the literature and the optimization solver, CPLEX. Regarding the lower bounds, the proposed methods found in almost all instances better results than those generated by the classical decomposition (per items) proposed in Toledo and Armentano (2006). Compared to CPLEX, the methods also proved to be very efficient since in most instances we found better lower bounds than the solver even after the branches.

Regarding the upper bounds, the results of the heuristics are better than those found by the Lagrangian heuristic proposed in Toledo and Armentano (2006) and are competitive with CPLEX.

For the lot sizing problem with setup crossover, some new formulations are proposed in order to explore ideas to avoid the necessity of defining new extra binary variables to model the setup crossover. Furthermore, we also propose symmetry breaking constraints for the formulations from the literature and quantify the value of the setup crossover in a computational experiment analyzing which factors influence this value. The results show that the proposed formulations are efficient. Regarding the benefits obtained allowing a setup crossover, we conclude that it depends on the characteristic of the problem, but especially for problems with high inventory costs and problems which allow the possibility of backlog cost it can be very significant.

Aiming to extend this work, we suggest as future research:

- To explore more the ideas of combining the techniques of Lagrangian relaxation and Dantzig-Wolfe decomposition, in order to improve and possibly extend the hybrid methods presented in this thesis;
- To apply the hybrid methods on other extensions of lot sizing problems, including integrated problems.
- To extend the proposed heuristics, to test new ideas to make feasible the demand constraints and to apply it to more difficult problems, such as, multi-stage or integrated problems;
- To propose algorithms (exact and/or heuristics) for the lot sizing problems with setup crossover;
- To extend the ideas of avoiding the necessity of defining extra binary variables to model the setup crossover for the problem with setup carryover and then propose new models for the problem with setup crossover and carryover.

Appendix A

Hybrid methods

In this appendix we will analyse the two different hybrid methods that combine Lagrangian relaxation and Dantzig-Wolfe decomposition to solve the linear relaxation of the master problem (MP_L) presented in Chapter 3. Note that Lagrangian relaxation and Dantzig-Wolfe decomposition have some advantages and disadvantages. The idea is to discuss how the relationship between these two techniques can be exploited to develop algorithms combining the strengths of these two methods.

First of all, consider the problem *(P)* defined by:

Compact Formulation:

$$z_P = min\ \ c^T x \tag{A.1}$$

subject to:

$$(P) \quad \begin{aligned} A^1 x &\leq b_1 \quad (Difficult) && \text{(A.2)}\\ A^2 x &\leq b_2 \quad (Easy) && \text{(A.3)}\\ x &\in Z_+^n && \text{(A.4)} \end{aligned}$$

wherein: $A^1 \in \Re^{m_1 \times n}$, $A^2 \in \Re^{m_2 \times n}$, $b_1 \in \Re^{m_1}$ e $b_2 \in \Re^{m_2}$.

Assume that the set of difficult constraints define the linking constraints and the set of easy constraints define the set which has a special structure. Let the set $S = \{x \in Z_+^n, A^2 x \leq b_2\}$. In this text, we will only consider the application of DW decomposition to the case where the set S is a limited polyhedron. Therefore, using discretization, see for example Nemhauser and Wolsey (1988), it is valid the following theorem:

Theorem A.1: Let $S = \{x \in Z_+^n, A^2 x \leq b_2\} \neq \emptyset$. Then there exists a finite set of extreme points $\{p_k\}_{k \in K} \subset conv(S)$ such that any point $x \in S$ can be represented as a convex combination of the finite number of extreme points of $conv(S)$, in other words:

$$x = \sum_{k=1}^{K} \lambda_k p_k \tag{A.5}$$

$$\sum_{k=1}^{K} \lambda_k = 1 \tag{A.6}$$

$$\lambda_k \geq 0 \; \forall k \in K \tag{A.7}$$

Where: K is the set of extreme points of $conv(S)$.

Thus, the problem P can be reformulated and now be represented in terms of extreme points by replacing x in its formulation. We obtain the following formulation, equivalent to the original problem, referred to as the master problem (MP):

Extended Formulation:

$$z_{MP} = min \; \sum_{k=1}^{K} (c^T p_k)\lambda_k \tag{A.8}$$

subject to:

$$\sum_{k=1}^{K} (A^1 p_k)\lambda_k \leq b_1 \quad (Difficult) \tag{A.9}$$

$$(MP) \quad \sum_{k=1}^{K} \lambda_k = 1 \tag{A.10}$$

$$\lambda_k \geq 0 \; \forall k \in K \tag{A.11}$$

The decision variables of the master problem are the λ_k $(k = 1, ..., K)$ variables and represent the weight of each extreme point p_k $(k = 1, ..., K)$. The difficult constraints $(A.9)$ define the linking constraints, now represented in terms of the extreme points and we have the convexity constraints $(A.10)$, which guarantee the choice of a convex combination of the extreme points of $conv(S)$.

Establishing the relationships between the linear relaxations of the problem MP_L and the problem P, usually the first one has an extraordinary high number of variables (equal to the number of points of the set K) when compared with second one, but substantially less constraints. However, the optimal solution of the master problem and consequently of the original problem can be determined without the explicit enumeration of all variables. To deal with the exponential number of variables, the authors usually use the method of column generation to solve the linear relaxation of the master problem.

Note that we could have used the discretization approach instead of convexification. However, as we are solving the linear relaxation of the master problem the distinction between these two approaches disappears (Vanderbeck and Wolsey, 2010).

The basic idea of column generation is: instead of considering all extreme points of the set *conv(S)* in the linear relaxation of the master problem (MP_L), we consider only a restricted set of these extreme points (amount enough to form a feasible inicial basis), thus defining a restricted master problem (RMP_L) and evaluate whether there are extreme points that are not currently in the RMP_L, which, if included in this problem could improve the value of the objective function.

Let π be the vector of dual variables associated with the constraints (A.9) and μ the dual variables associated with the constraints (A.10). These dual variables are obtained by solving the problem RMP_L, for example, using the simplex method. The reduced cost of a variable λ_k is given by: $cp_k - \pi A^1 p_k - \mu$. Therefore, if $cp_k - \pi A^1 p_k - \mu \geq 0, \forall k \in K$, then the optimal solution of the RMP_L, is also the optimal solution of the master problem MP_L. If from the set of all variables that do not belong to RMP_L, there is at least one with negative reduced cost, then this column should be added to RMP_L, and the process must be repeated.

To find the λ_k variable that has the most negative reduced cost, we solve the following problem:

$$z_{SUB} = min \ \ (c + A^1\pi)x - \mu \tag{A.12}$$

subject to:

$$(SUB) \quad A^2 x \leq b_2 \quad (Easy) \tag{A.13}$$

$$x \in Z_+^n \tag{A.14}$$

This problem is usually referred to subproblem (SUB). Thus, if $z_{SUB} \geq 0$, the optimal solution of the MP_L was found. If the value of $z_{SUB} < 0$, a new column should be added to RMP_L and the procedure presented should be repeated until there are no more attractive columns.

A.1 Lagrangian relaxation applied to the extended formulation

The dual solutions used to generate columns have a key role in the performance of the column generation method. The standard strategy, in other words, to use the simplex method to solve the RMP_L, and then obtain optimal dual variables usually cause an unstable behavior in the method, specially when these variables are extreme points of the dual set. This typically leads to a large number of iterations and slow convergence in the

last iterations of the method. It can occurs because these solutions usually vary greatly from one iteration to another (Briant et al., 2008; Ben Amor et al., 2009; Gondzio et al., 2013).

Figure A.1 illustrates the oscillation of the optimal dual solution throughout four iterations of the column generation method using the standard strategy. The figure is split in four parts, each one representing the dual feasible set of the corresponding RMP_L. In part (a) of Figure A.1 we represent the initial dual feasible set, the respective optimal solution represented by the ball in the bottom of the figure and the corresponding optimal value represented by the dotted line. This dual optimal solution is sent to the subproblems, which generate a new column (represented as a new cut in the dual feasible set). In part (b), we have the new feasible dual set which is obtained by adding this new column to RMP_L. The parts (c) and (d) of the Figure A.1 follows the same explanation considering the next iterations of the method. The purpose of this illustration is to show the unstable behavior of the optimal dual solutions obtained during the iterations. After adding the columns generated to RMP_L the dual solutions "jump" from one side to the other of the dual feasible set and beyond this oscillation, we observe that the values of the solutions does not change a lot, causing a slow convergence of the method.

Because of this behavior, different techniques for stabilizing dual solutions have been proposed in the literature (see, for exemple Gondzio et al. (2013)). It leads to many variations of the column generation method. These techniques are based on the fact that any dual feasible solution (not just the optimal solution) can be used to generate columns for the RMP_L.

Following this trend, the idea of this method is instead of using the simplex method to obtain the dual variables of the linear relaxation of the extended formulation (A.8)-(A.11) (MP_L problem) we can use Lagrangian relaxation to approximate these values.

Formally, to solve approximately the linear programming problem MP_L the difficult constraints (A.9) are dualized in the objective function (A.8):

$$z_{MP_L}(\lambda) = min \sum_{k=1}^{K} (cp_k)\lambda_k - \pi(b_1 - \sum_{k=1}^{K} (A^1 p_k)\lambda_k) \tag{A.15}$$

$RL(\lambda)$ subject to:

$$\sum_{k=1}^{K} \lambda_k = 1 \tag{A.16}$$

$$\lambda_k \in [0,1], \forall k \in C \tag{A.17}$$

In general the problem (A.15)-(A.17) can be decomposed and easily solved. The optimal Lagrange multipliers π^* can be approximated using the subgradient optimizations

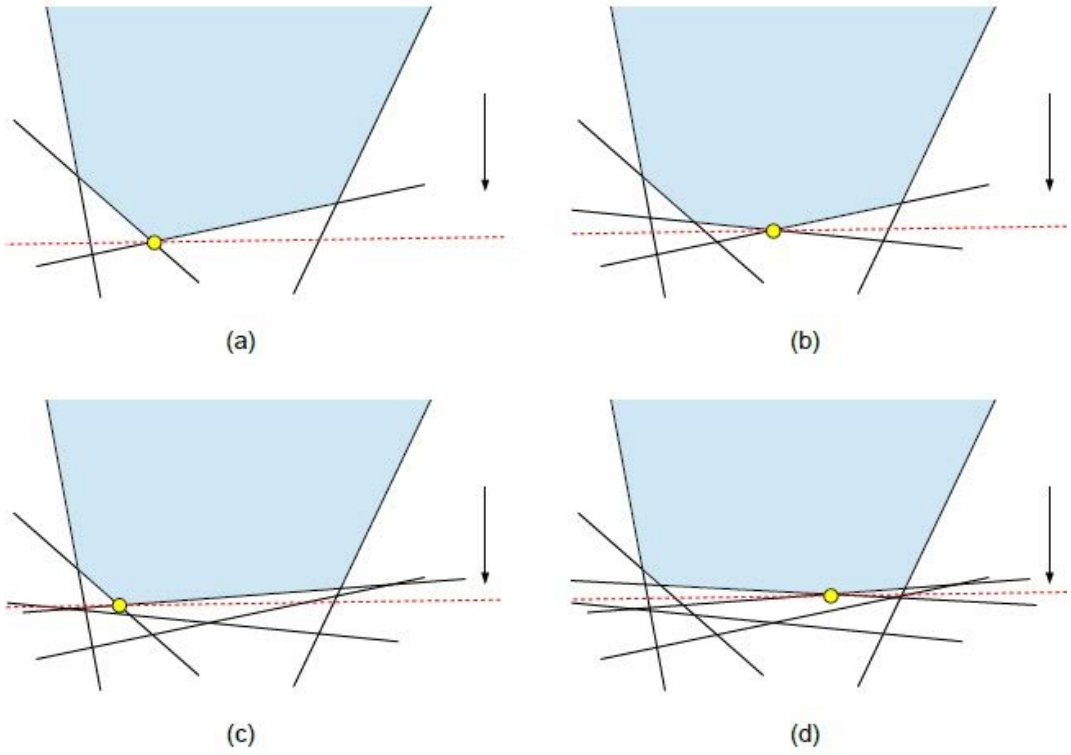


Figure A.1: Illustration of the unstable behavior of the optimal dual solutions (figure presented originally in Munari (2013)).

method. This method is an iterative process that through an initial set of Lagrange multipliers, generates a sequence of multipliers whose limit tends to the solution of the Lagrangian dual problem. Following, we will present a summary of subgradient algorithm:

Subgradient Algorithm

The gradient method is well known in the literature and is used to optimize functions that are differentiable at all points of the domain. By definition, the gradient of a function in a point x is given by:

$$\nabla f(x) = (\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n})$$

and indicates in which direction the function $f(x)$ grows. Each component $\frac{\partial f(x)}{\partial x_i}$ represents the partial derivative of the function $f(x)$ with respect to x_i at the point x. For minimization problems with objective function $f(x)$, $-\nabla f(x)$ indicates where this function decreases.

For a fixed point x and a direction d such that $-\nabla f(x).d > 0$, the idea of the gradient optimization method is based on the fact that a small step of value t, given in direction

d yields a solution with a smaller value for the objective function, i.e., using a point $y = x + td$, we have that $f(y) < f(x)$.

The subgradient optimization method is an adaptation of the gradient method for functions that are not differentiable at all points of the domain. For dual minimization problems, the idea of the method is to consider steps, t, in the negative direction of a subgradient so that the values of relaxation, that form a convex function, decrease at each iteration. Before presenting the method, we will define subgradient and subdifferential.

Definition A.1 *A vector $\gamma \in \Re^n$ is a subgradient of a function $f : \Re^n \to \Re$ if $f(x) \leq f(x^0) + \gamma^T(x - x^0)$ for all $x \in \Re^n$.*

If the function f is differentiable at x^0, then we have only one subgradient which is equal to the gradient of f in x^0.

Definition A.2 *The subdifferential $\partial f(x^0)$ of f in x^0 is the set of all subgradient f in x^0.*

If f is differentiable in x^0, then $\partial f(x^0) = \{\nabla f(x^0)\}$, in which $\nabla f(x^0)$ represents the gradient of f in x^0. Note also that $\partial z_{RL}(\lambda) \neq \emptyset$.

The subgradient method provides a multiplier λ^k for the current relaxation in each iteration k, starting from a initial vector λ^0 (for example, the null vector). As previous discussed, we should update each current multiplier in the opposite direction to a current subgradient, $s(\lambda^k)$, according to a given step, t_k, for example:

$$\lambda^{k+1} = \lambda^k + t_k s(\lambda^k)$$

where

$$t_k = \frac{[\overline{z_P} - z_{RL}(\lambda^k)]\rho^k}{\| s(\lambda)^k \|^2}$$

The value $\overline{z_p}$ can be obtained by calculating the objective function obtained by the application of a heuristic, i.e., a feasible solution for P; and $z_{RL}(\lambda^k)$ is the objective function value of the relaxation, that was solved with the vector λ^k. Therefore, $\overline{z_p} - z_{RL}(\lambda^k))$ is the difference between an upper and a lower bound for the primal problem P. The parameter ρ^k adjusts the size of the step.

Note that for $s(\lambda)^k$ be a valid subgradient and point to where the objective function of the relaxation decreases, $\overline{z_p}$ must be a concave function on a dual maximization problem (the relaxation and the primal problem must therefore be a minimization problem). Finally, also note that variations of the subgradient method can be found in the literature.

Proposition A.1: With the approximation to the vector π^*, it is possible to calculate an approximation to the vector μ^* that represent the optimal multiplier regarding the convex constraints (A.10) using the formula $\mu = c - A^1\pi$.

Proof: First of all, note that we can denote the coefficient matrix of the problem

(A.8)-(A.11) and the dual variables of the constraints (A.9) and (A.10) by: $B = \begin{bmatrix} A^1 \\ 1 \end{bmatrix}$ and $\eta = [\pi \quad \mu]$, respectively.

To calculate the dual variables, it is well known the we have to solve the system $\eta B = c$. Therefore:

$$\eta B = c \Rightarrow [\pi \quad \mu] \begin{bmatrix} A^1 \\ 1 \end{bmatrix} = c \Rightarrow \pi A^1 + \mu = c \Rightarrow \mu = c - \pi A^1 \blacksquare$$

The approximated Lagrangian multipliers can be used in the subproblem (A.12)-(A.14) to generate new columns that are added to RMP_L and, in a next step the optimal dual variables π^* and μ^* for the updated RMP_L are approximate again by Lagrangian relaxation.

For each approximation to the solution of the restricted master problem RMP_L, a lower bound for the MP_L can be calculated by adding the value of z_{MP_L} with z_{SUB}.

Next, in Figure A.2, we present a generic flow diagram that illustrates the application of Lagrangian relaxation in the extended formulation to obtain a lower bound for a problem P.

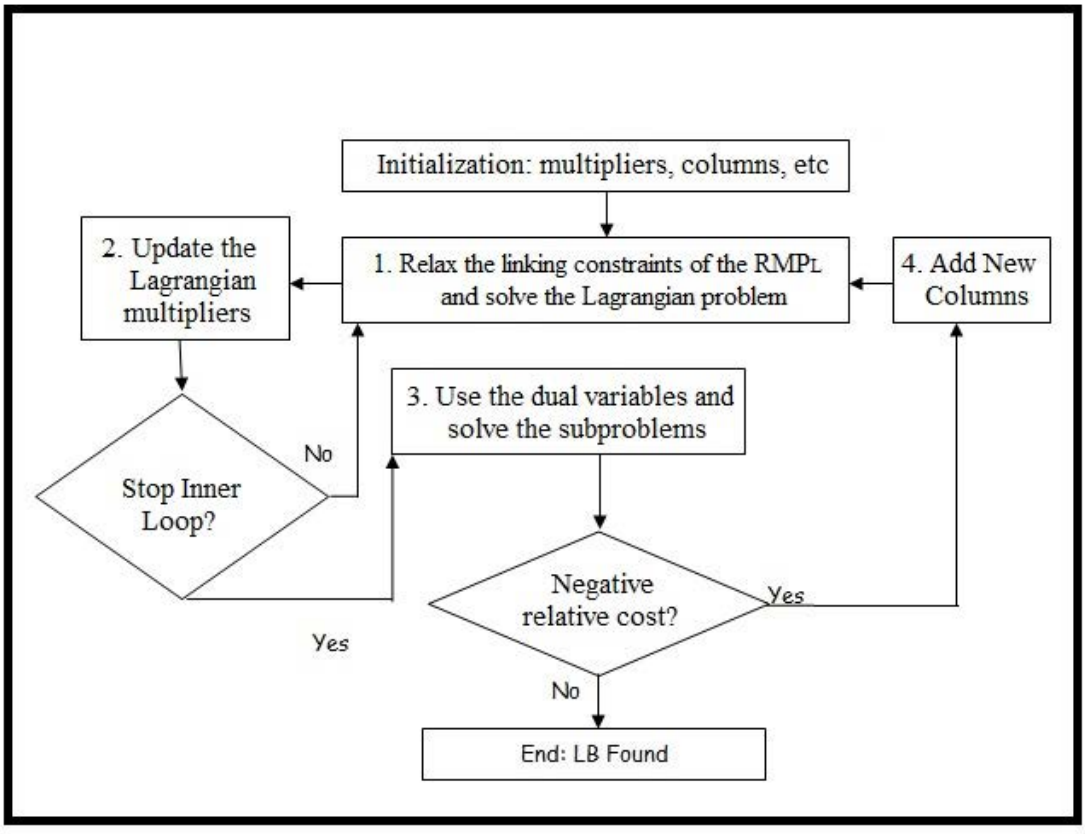


Figure A.2: Lagrangian relaxation in the extended formulation

A.2 Lagrangian relaxation applied to the extended and the compact formulation

This method iterates between column generation and Lagrangian relaxation. The idea is to use the Lagrangian relaxation in the extended formulation, which is computationally less expensive than the simplex method to solve the restricted master problem (as in the previous method) and Lagrangian relaxation in the compact formulation.

Considering the compact formulation (problem P given by $(A.1) - (A.4)$), the idea of Lagrangian relaxation is to relax the difficult constraints $(A.2)$ placing them in the objective function as a "penalty". Thus, the Lagrangian relaxation of the compact formulation with respect to the difficult set of constraints is defined by associating to this set a vector $\lambda \geq 0$, called Lagrange multiplier or dual variables. The Lagrangian problem obtained using the technique of Lagrangian relaxation to the compact formulation is given by:

$$z_{RL}(\lambda) = min\{z(\lambda, x) = c^T x + \lambda^T (A^1 x - b_1)\} \tag{A.18}$$

$RL(\lambda)$ subject to:

$$x \in S_{LR} = \{x \in Z_+^n : A^2 x \leq b_2\} \tag{A.19}$$

where $\lambda \in \Re_+^{m1}$

Note that the subproblems resulting from Dantzig-Wolfe decomposition (A.12)-(A.14) and the Lagrangian problem (A.18)-(A.19) are the same unless a constant in the objective function. Therefore, the main idea of this method is to use Lagrangian relaxation in the compact formulation to generate columns to the RMP_L, in other words, the solutions generated by Lagrangian relaxation can be added as new columns to the restricted master problem.

Therefore, with this method we would like to achieve the following objectives: to avoid an unstable behavior in the column generation method (degeneration); converge quickly by adding blocks of columns generated by Lagrangian relaxation.

Figure A.3, presents a generic flow diagram that illustrates the application of this method.

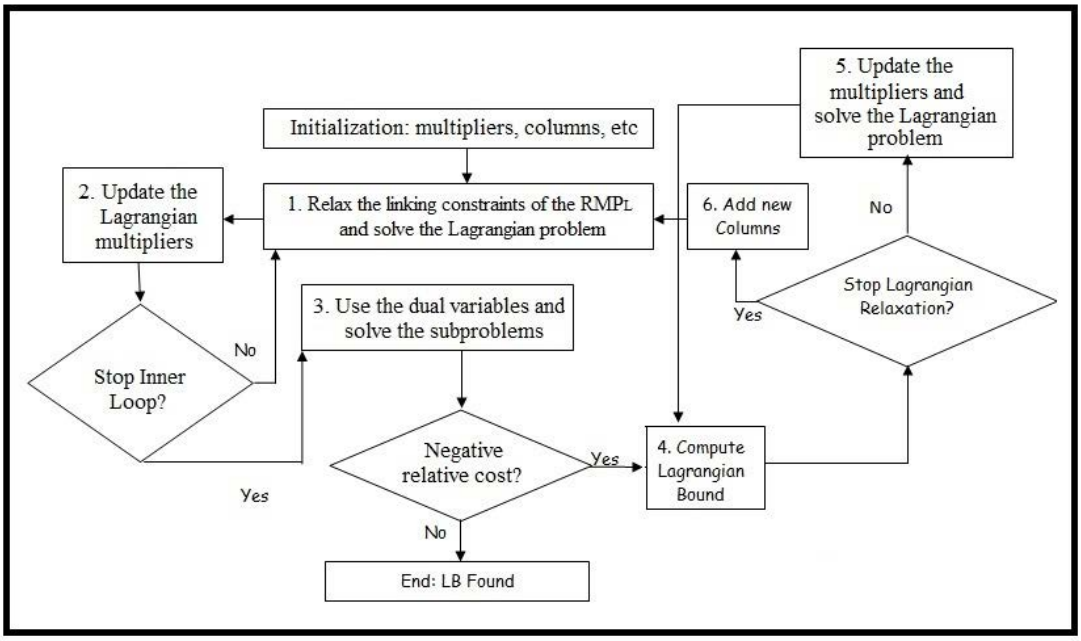


Figure A.3: Lagrangian relaxation in the extended and compact formulation

Bibliography

[Akartunali and Miller (2012)] K. Akartunali, A. J. Miller, A computational analysis of lower bounds for big bucket production planning problems. *Computational Optimization and Applications*, **53** (2012), 729–753.

[Alfieri et al. (2002)] A. Alfieri, P. Brandimarte, S. D'Orazio, LP-based heuristics for the capacitated lot-sizing problem: The interaction of model formulation and solution algorithm. *International Journal of Production Research*, **40** (2002), 441–458.

[Andriolo et al. (2014)] A. Andriolo, D. Battini, A. Persona, F. Sgarbossa, R. W. Grubbström, A Century of evolution from harris's basic lot size model: survey and research agenda. *Int. J. Production Economics*, **155** (2014), 16–38.

[Araujo and Clark (2013)] S. A. Araujo, A. Clark, A priori reformulations for joint rolling-horizon scheduling of materials processing and lot-sizing problem. *Computers & Industrial Engineering*, **65** (2013), 577–585.

[Araujo et al. (2015)] S. A. de Araujo, B. De Reyck, Z. Degraeve, I. Fragkos, R. Jans, Period decompositions for the capacity constrained lot size problem with setup times. *INFORMS Journal on Computing*, (2015) In press.

[Bahl (1983)] H. C. Bahl, Column generation based heuristic algorithm for multi-item scheduling. *IIE Trans*, **15** (1983), 136–141.

[Barnhart et al. (1998)] C.E.L. Barnhart, G.L. Johnson, M.W.P. Nemhauser, P. H. Vance, Branch-and-price: column generation for solving huge integer programs. *Operations Research*, **46** (1998), 316–329.

[Belo-Filho et al. (2014)] M. A. F. Belo-Filho, B. Almada-Lobo, F. M. B. Toledo, Models for capacitated lot-sizing problem with backlogging, setup carryover and crossover, *Journal of the Operational Research Society*, **65 (11)** (2014), 1735–1747.

[Belvaux and Wolsey (2000)] G. Belvaux, L. A. Wolsey, Bc-prod: a specialized branch-and- cut system for lot-sizing problems. *Management Science*, **46** (2000), 993–1007.

[Ben Amor et al. (2007)] H. Ben Amor, J. Desrosiers, J.M. Valèrio de Carvalho, Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54** (2007), 454–463.

[Ben Amor et al. (2009)] H. Ben Amor, J. Desrosiers, A. Frangioni, On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, **157 (6)** (2009), 1167-1184.

[Bitran and Matsuo (1986)] G. R. Bitran, H. Matsuo, The multi-item capacitated lot size problem: error bounds of Manne's formulations. *Management Science*, **32** (1986), 350–359.

[Bixby et al. (1992)] R. E. Bixby, J. W. Gregory, I. J. Lustin, R. E. Marsten, D. F. Shanno, Very large-scale linear programming: a case study in combining interior point and simplex methods. *Operations Research*, **40** (1992), 885–897.

[Brahimi et al. (2006)] N. Brahimi, D. S. Peres, N. M. Najid, A. Nordli, Single item lot sizing problem. *European Journal of Operational Research*, **168** (2006), 1–16.

[Briant et al. (2008)] O. Briant, C. Lemarechal, P. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck, Comparison of bundle and classical column generation. *Mathematical programming*, **113 (2)** (2008), 299-344.

[Camargo et al. (2012)] V. C. B. Camargo, F. M. B. Toledo, B. Almada-Lobo, Three time-based scale formulations for the two-stage lot sizing and scheduling in process industries. *Journal of the Operational Research Society*, **63** (2012), 1613–1630.

[Camerini et al. (1975)] P. M. Camerini, L. Fratta, F. Maffioli, On improving relaxation methods by modified gradient techniques. *Mathematical Programming Study*, **3** (1975), 26–54.

[Caserta and Voss (2013)] M. Caserta, S. Voss, A math-heuristic Dantzig-Wolfe algorithm for capacitated lot sizing. *Annals of Mathematics and Artificial Intelligence*, **69 (2)** (2013), 207–224.

[Carreno (1990)] J. J. Carreno, Economic lot scheduling for multiple products on parallel identical processors. *Management Science*, **36** (1990), 348–358.

[Cattrysse et al. (1990)] D. Cattrysse, J. Maes, L. N. van Wassenhove, Set partitioning and column generation heuristics for capacitated dynamic lotsizing. *European Journal of Operational Research*, **46** (1990), 38–47.

[Cattrysse et al. (1993)] D. G. Cattrysse, M. Salomon, R. Kuik, V. L. N. Wassenhove, A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times. *Management Science*, **39** (1993), 447–486.

[Chen and Thizy (1990)] W. H. Chen, J. M. Thizy, Analysis of relaxation for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, **26** (1990), 29–72.

[Dantzig and Wolfe (1960)] G. B. Dantzig, P. Wolfe, Decomposition principles for linear programming. *Operations Research*, **8** (1960), 101–111.

[Degraeve and Jans (2007)] Z. Degraeve, R. Jans, A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacited lot-sizing problem with setup times. *Operations Research*, **55** (2007), 909–920.

[De Matta and Guignard (1994)] R. De Matta, M. Guignard, Dynamic production scheduling for a process industry. *Operations Research*, **42** (1994), 492–503.

[De Matta and Guignard (1995)] R. De Matta, M. Guignard, The performance of rolling production schedules in a process industry. *IIE Transactions*, **27** (1995), 564–573.

[Denizel and Süral (2006)] M. Denizel, H. Süral, On alternative mixed integer programming formulations and LP based heuristics for lot-sizing with setup times. *Journal of the Operational Research Society*, **57** (2006), 389–399.

[Denizel et al. (2008)] M. Denizel, F. T. Altekin, H. Süral, H. Stadtler, Equivalence of the lp relaxation of two strong formulation for the capacitated lot-sizing problem with setup times. *OR spectrum*, **30** (2008), 773–785.

[Diaby et al. (1992)] M. Diaby, H. Bahl, M. H. Karwan, S. Ziont, Capacitated lot-sizing and scheduling by lagrangean relaxation, *European Journal Of Operational Research*. **59** (1992), 444–458.

[Dzielinski and Gomory (1965)] B. P. Dzielinski, R. E. Gomory, Optimal programming of lot sizes inventory and labor allocations. *Management Science*, **11** (1965), 874–890.

[Eppen and Martin (1987)] G. B. Eppen, R. K. Martin, Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, **6** (1987), 832–848.

[Fandel and Stammen-Hegener (2006)] G. Fandel, C. Stammen-Hegener, Simultaneous lot sizing and scheduling for multi-product multi-level production. *International Journal of Production Economics*, **104** (2006), 308–316.

[Fiorotto and Araujo (2012)] D. J. Fiorotto, S. A. de Araujo, Relaxação lagrangiana aplicada ao problema de dimensionamento de lotes em máquinas paralelas: Limitantes inferiores. *TEMA - Têndencias em Matemática Aplicada e Computacional*, **13** (2012), 13–24.

[Fiorotto and Araujo (2014)] D. J. Fiorotto, S. A. de Araujo, Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines. *Annals of Operations Research*, **217** (2014), 213–231.

[Gondzio et al. (2013)] J. Gondzio, P. G. Brevis, P. Munari, New developments in the primal dual column generation technique. *European Jounal of Operational Research*, **224** (2013), 41–51.

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

[Haase (2005)] K. Haase, Solving large-scale capacitated lot-sizing problems close to optimality. *Working paper, Technische Universität Dresden*, (2005).

[Helber and Sahling (2010)] S. Helber, F. Sahling, A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *Journal Production Economics*, **123** (2010), 247–256.

[Hindi (1995)] K. S. Hindi, Computationally efficient solution of the multi-item capacitated lot-sizing problem. *Comput Ind Eng*, **28** (1995), 709–719.

[Hindi (1996)] K. S. Hindi, Solving the CLSP by a tabu search heuristic. *European Journal of Operational Research*, **47** (1996), 151–161.

[Huisman et al. (2005)] D. Huisman, R. Jans, M. Peeters, A. P. M. Wagelmans, Combining column generation and lagrangian relexation. *G. Desaulniers, J. Desrosiers, M. Solomon, eds. Column Generation. Springer, New York*, (2005), 247–270.

[Jans (2009)] R. Jans, Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. *INFORMS Journal on Computing*, **21** (2009), 123–136.

[Jans and Degraeve (2004a)] R. Jans, Z. Degraeve, Improved lower bounds for capacitated lot sizing problem with setup time. *Operation Research Letters*, **32** (2004a), 185–195.

[Jans and Degraeve (2004b)] R. Jans, Z. Degraeve, An industrial extension of the discrete lot sizing and scheduling problem. *IIE Transactions*, **36** (2004b), 47–58.

[Jans and Degraeve (2007)] R. Jans, Z. Degraeve, Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, **177** (2007), 1855–1875.

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

[Jans and Desrosiers (2013)] R. Jans. J. Desrosiers, Efficient symmetry breaking formulations for the job grouping problem. *Computers and Operations Research*, **40(4)** (2013), 1132–1142.

[Kaczmarczyk (2009)] W. Kaczmarczyk, Modelling multi-period set-up times in the proportional lot-lizing problem. *Decision Making in Manufacturing and Services*, **3** (2009), 15–35.

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

[Kopanos et al. (2011)] G. M. Kopanos, L. Puigjaner, C. T. Maravelias, Production planning and scheduling of parallel continuous processes with product families. *Industrial and Engineering Chemistry Research*, **50(3)** (2011), 1369–1378.

[Krarup and Bilde (1977)] J. Krarup, O. Bilde, Plant location, set covering and economic lot size: An O(mn)-algorithm for structured problems. *Numerische Methoden bei Optimierungsaufgaben. Bang 3: Optimierung bei Graphentheoritischen Ganzzahligen Problemen*, (1977), 155–186.

[Lambrecht and Vanderveken (1979)] M. Lambrecht, H. Vanderveken, Heuristic procedures for the single operation, multi-item loading problem. *AIIE Transactions*, **11** (1979), 319–326.

[Lasdon and Terjung (1971)] L. S. Lasdon, R. C. Terjung, An efficient algorithm for multi-item scheduling. *Operations Research*, **19** (1971), 946–969.

[Manne (1958)] A. S. Manne, Programming of economic lot sizes. *Management Science*, **4** (1958), 115–135.

[Marinelli et al. (2007)] F. Marinelli, M. E. Nenni, A. Sforza, Capacitated lot sizing and scheduling with parallel machines and shared buffers: A case study in a packaging company. *Annals of Operations Research*, **150**, (2007), 177–192.

[Menezes et al. (2010)] A. A. Menezes, A. Clark. B, Almada-Lobo, Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. *Journal of Scheduling*, **14(2)** (2010), 209–219.

[Mergaux and van Wassenhove (1984)] L, P. Mergaux, L. N. van Wassenhove, Production planning with capacity constraints, Master thesis, Division of Industrial Management, Katholieke Universiteit Leuven, Dutch, 1984.

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

[Meyr and Mann (2013)] H. Meyr, M. Mann, A decomposition approach for the general lotszing and scheduling problem for parallel production lines. *European Journal of Operational Research*, **229** (2013), 718–731.

[Mohan et al. (2012)] S. Mohan, M. Gopalakrishnan, R. Marathe, A. Rajan, A note on modelling the capacitated lot-sizing problem with set-up carryover and set-up splitting. *International Journal of Production Research*, **50(19)** (2012), 5538–5543.

[Munari (2013)] P. Munari, Theoretical and computational issues for improving the performance of linear optimization methods. Tese (Doutorado), Instituto de Ciências Matemáticas e de Computação-ICMC, Universidade de São Paulo, São Carlos, (2013).

[Nemhauser and Wolsey (1988)] G. L. Nemhauser, L. A. Wolsey, Integer and combinatorial optmization, John Wiley & Sons, (1988).

[Ozdamar and Birbil (1998)] L. Ozdamar, S. I. Birbil, Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. *European Journal of Operational Research*, **110** (1998), 525–547.

[Pimentel et al. (2010)] C. M. O. Pimentel, F. P. Alvelos, J. M. V. Carvalho, Comparing Dantzig-Wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lot sizing problem. *Optimization Methods and Software*, **25** (2010), 229–319.

[Pochet and van Vyve (2004)] Y. Pochet, M. van Vyve, A generic heuristic for production planning problems. *INFORMS Journal of Computing*, **16**, 316–327 (2004).

[Salomon et al. (1991)] M. Salomon, L. G. Kroon, R. Kuik, L. N. van Wassenhove, Some extensions of the discrete lot-sizing and scheduling problem. *Management Science*, **37** (1991), 801–812.

[Salomon et al. (1993)] M. Salomon, R. Kuik, L. N. van Wassenhove, Statistical search methods for lotsizing problems. *Operations Research*, **41** (1993), 453–468.

[Sherali and Smith (2001)] H. D. Sherali, J. C. Smith, Improving discrete model representations via symmetry considerations. *Management Science*, **47(10)** (2001), 1396–1407.

[Sinha and Zoltners (1979)] P. Sinha, A. A. Zoltners, The multiple-choice knapsack problem. *Operations Research*, **27 (3)** (1979), 503–515.

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

[Stadtler (2003)] H. Stadtler, Multilevel lot sizing with set up times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, **51** (2003), 487–502.

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

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

[Sung (1986)] C. S. Sung, A single-product parallel-facilities production-planning model. *International Journal of Systems Science*, **17** (1986), 983–989.

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

[Süral et al. (2009)] H. Süral, M. Denizel, L. N. van Wassenhove, Lagrangean relaxation based heuristic for lot sizing with setup times. *European Journal of Operational Research*, **195** (2009), 51–63.

[Tempelmeier and Buschkuhl (2009)] H. Tempelmeier, L. Buschkuhl, A heuristic for the dynamic multi-level capacitated lotsizing problem with linked lotsizes for general product structures. *Or Spectrum*, **31** (2009), 385–404.

[Thizy and van Wassenhove (1985)] J. M. Thizy, L. N. van Wassenhove, Lagrangean relaxation for the multi-item capacitated lot-sizing problem: a heuristic implementation. *AIIE Transactions*, **17** (1985), 64–74.

[Toledo (1998)] F. M. B. Toledo, Dimensionamento de Lotes em Máquinas Paralelas. Tese (Doutorado), Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas, Campinas, (1998).

[Toledo and Armentano (2006)] F. M. B. Toledo, V. A. Armentano, A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research*, **175** (2006), 1070–1083.

[Trigeiro et al. (1989)] W. W. Trigeiro, J. Thomas, J. O. McClain, Capacitated lot sizing with setup times. *Management Science*, **35** (1989), 353–366.

[Vanderbeck and Wolsey (2010)] . Vanderbeck, L. A. Wolsey, Reformulation and decomposition of integer programs. *Jünger, M.; Liebling, T. M.; Naddef, D.; Nemhauser, G. L.; Pulleyblank, W. R.; Reinelt, G.; Rinaldi, G.; Wolsey, L. A., eds. 50 Years of Integer Programming 1958-2008*, Springer Berlin Heidelberg, (2010), 431–502.

[Wagner and Whitin (1958)] H. M. Wagner, T. M. Whitin, Dynamic version of the economic lotsize model. *Management Science*, **5** (1958), 89–96.

[Wu and Shi (2011)] T. Wu, L. Shi, Mathematical models for capacitated multi-level production planning problems with linked lot sizes. *International Journal of Production Research*, **49 (20)** (2011), 6227–6247.