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

Eduardo Machado Silva

The One-dimensional Multi-Period Cutting Stock Problem with Setup Cost

São José do Rio Preto
2023

Eduardo Machado Silva

The One-dimensional Multi-Period Cutting Stock Problem with Setup Cost

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", Câmpus de São José do Rio Preto.

Financiadora: CAPES

Orientador: Prof. Dr. Silvio Alexandre de Araujo

São José do Rio Preto

2023

S586o

Silva, Eduardo Machado

The One-dimensional Multi-Period Cutting Stock Problem with Setup Cost / Eduardo Machado Silva. -- São José do Rio Preto, 2023

123 p. : il., tabs.

Tese (doutorado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Silvio Alexandre de Araujo

1. Matemática. 2. Pesquisa Operacional. 3. Otimização Matemática. 4. Programação inteira-mista. 5. Problemas de corte de estoque multiperíodo. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

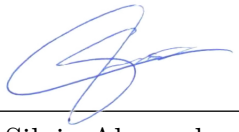
Eduardo Machado Silva

The One-dimensional Multi-Period Cutting Stock Problem with Setup Cost

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", Câmpus de São José do Rio Preto.

Financiadora: CAPES

Comissão Examinadora



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

Prof(a). Dr(a). Kelly Cristina Poldi
UNICAMP - Campinas

Prof. Dr. Horacio Hideki Yanasse
UNIFESP - São José dos Campos

Prof. Dr. Diego Jacinto Fiorotto
UNICAMP - Limeira

Prof. Dr. Antônio Augusto Chaves
UNIFESP - São José dos Campos

São José do Rio Preto
26 de Janeiro de 2023

AGRADECIMENTOS

Foram 11 anos do início dessa trajetória, sou grato a todos que fizeram parte e me ajudaram a chegar até esse momento. Grato aos professores do ensino fundamental e médio de Cassilândia pelo alicerce básico para eu inicar meu curso de graduação. Grato aos professores do curso de Matemática da UEMS-Cassilândia pelo alicerce da minha trajetótia acadêmica, em especial aos professores Marco e Regina por terem me orientado durante essa etapa. Sou grato ao professor Maurílio por ter sido um excelente orientador e ter iniciado a minha trajeória no IBILCE quando aceitou me orientar durante mestrado.

Agradeço ao excelente corpo docente do IBILCE por todos os ensinamentos nesse imenso universo das teorias da matemática. Um agradecimento especial ao meu orientador de doutorado, professor Silvio, por ter sido um excelente orientador, ter aceitado me orientar mesmo vindo de outra área, por todas as orientações (não apenas acadêmicas) e paciência durante esses longos 5 anos.

Sou grato ao professor Kerem por ter aceitado me supervisionar durante meu estágio no exterior e além de tudo por todo apoio, conversas e passeios durante o meu 2020 em Glasgow. Além de um excelente supervisor foi um psicólogo nas horas vagas (risos). Também sou grato ao professor Raf pela supervisão no Canadá.

Agradeço a minha família pelo apoio emocional durante toda minha trajetória, em especial aos meus avós. Agradeço aos meus amigos que conquistei do ensino médio ao doutorado, pelos momentos de descontração, conversas, apoio e por me aturarem até o presente momento (risos). A uma lista de pessoas que compartilhei momentos alegres, sou grato.

Agradeço a Gislaine pelo apoio durante meus anos inicias de doutorado, pelas dis-

cussões e códigos disponibilizados. Sou grato ao Thiago por todo apoio técnico durante esses 5 anos. Também sou grato a equipe responsável pelo restaurante universitário do IBILCE pelas deliciosas refeições. Nada como um almoço muito bem feito após uma manhã de correria.

Agradeço a banca examinadora deste trabalho que contribuiu com a melhoria do mesmo. Obrigado professora Kelly e professores, Washington, Horacio, Chaves e Diego.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

Nesta tese o problema de corte de estoque unidimensional multiperíodo com minimização dos custos de preparação nos padrões de corte é estudado. Primeiro propomos um extenso conjunto de formulações pseudo-polinomiais e baseadas em padrões de corte, principalmente adaptando formulações conhecidas para problemas de corte de estoque da literatura. Reformulações baseadas no problema de localização de facilidades são discutidas para melhorar os limitantes inferiores dos modelos propostos. Em seguida, apresentamos uma análise teórica comparando as várias formulações propostas em relação ao seu limitante inferior. Apresentamos uma análise computacional a fim de complementar a análise teórica e apresentar mais *insights* com relação à complexidade e qualidade das formulações na prática. Os experimentos computacionais foram realizados em dois conjuntos de instâncias sendo o segundo mais difícil de ser resolvido. Ambos os conjuntos de instâncias mostraram que as reformulações baseadas no problema de localização de facilidade propostas melhoram a qualidade dos limitantes inferiores. Contudo, testes adicionais mostram que a melhoria do limitante é afetada quando maiores custos do objeto em estoque são considerados. Em uma abordagem diferente, um algoritmo genético de chaves aleatórias viciadas em que o controle dos parâmetros é adaptativo é proposto para resolver o problema. Para a inicialização da metaheurística, um procedimento de decodificação das chaves aleatórias em termos da solução (*decoder*) do problema é necessário. Dois (*decoders*) são propostos e avaliados com base nos resultados de uma heurística de geração de colunas integrada a um *software* de solução. Uma combinação da metaheurística e da heurística de geração de colunas também é apresentada. Os resultados computacionais mostram que ambos os processos de decodificação obtém um melhor desempenho que a heurística de geração de colunas para instâncias com items pequenos cujo custo de preparação nos padrões de corte é maior em relação ao custo do objeto em estoque. Por fim, são discutidas as conclusões finais e propostas de pesquisa futura.

Palavras chave: Problema de Corte de Estoque Multiperíodo. Setup nos Padrões de Corte. Formulações Fortes. Algoritmo Genético Adaptativo com Chaves Aleatórias Viciadas (*ABRKGA*).

ABSTRACT

In this thesis, we study the multi-period one dimensional cutting stock problem minimizing setup costs on cutting patterns. We first propose an extensive range of pattern-based and pseudo-polynomial formulations for the problem, primarily adapting known formulations for cutting stock problems from the literature. Facility location based reformulations are also discussed to improve the lower bounds of the proposed models. We then present a thorough theoretical analysis to establish the strength of the various proposed formulations in comparison to each other. A computational analysis is presented to complement the theoretical analysis and present further insights with respect to the complexity and strength of the formulations in practice. The computational experiments were performed over two sets of instances where the second set is more difficult to solve. Both sets of instances have shown that the proposed facility location reformulations significantly improve the quality of the lower bounds. However, additional computational tests show that the lower bound improvement is directly affected by the object cost. Then, a random key genetic algorithm with adaptive parameter control is proposed to solve the problem. The metaheuristic initialization requires a decoder process which maps the random keys to feasible solutions of the problem. Two different decoder processes are proposed and evaluated according to the performance of a column generation heuristic solved by an optimization software. An experiment combining both the metaheuristic and the column generation is also presented. Computational experiments show that both decoders outperform the column generation based heuristic for instances with small items length and setup cost greater than the object cost. Finally, some final conclusions and future research are discussed.

Keywords: Multi-Period Cutting Stock Problem. Cutting Pattern Setup. Strong Formulations. Adaptive Biased Random Keys Genetic Algorithm (*ABRKGA*).

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OR	Operational Research;
CSP	Cutting Stock Problem;
PMP	Pattern Minimization Problem;
CSPs	Cutting Stock Problem with setups on cutting patterns;
$\mathcal{NP}$-hard	Non-deterministic polynomial-time hard;
MPCSP	Multi-Period Cutting Stock Problem;
MPCSPs	Multi-Period Cutting Stock Problem with Setup Costs;
VRP	Vehicle Routing Problem;
AGG	Adapted Gilmore and Gomory;
AJS	Adapted Johnston and Sadinlija;
ARE	Adapted Reflect;
AVR	Adapted Vehicle Routing;
AGGFL	Adapted Gilmore and Gomory Facility Location;
AJSFL	Adapted Johnston and Sadinlija Facility Location;
AREFL	Adapted Reflect Facility Location;
AVRFL	Adapted Vehicle Routing Facility Location;
EA	Evolutionary Algorithm;
GA	Genetic Algorithm;
RKGA	Random Key Genetic Algorithm;
BRKGA	Biased Random Key Genetic Algorithm;
ABRKGA	Adaptive Biased Random Key Genetic Algorithm;

CONTENTS

CHAPTER 1

INTRODUCTION

Operational Research (*OR*) is one of the managerial decision science tools used by profit and non-profit organizations. As the global environment becomes fiercely competitive, OR has gained significance in applications such as green logistic, quality management, benchmarking and decision making techniques. The growth of global markets and the resulting increase in competition have highlighted the need for *OR*. In order to be competitive, business must meet the challenges present in a global market by offering products and services with good value to their costumers (Agrawal et al. (2010)). *OR* leads to a more efficient use of resources, which is not only cost attractive, but lead to environment friendly decisions as well, being then an essential management tool to gain competitiveness in a industrial environment (Dekker et al. (2012)).

Among the very first ideas to emerge from *OR* to be applied in practice, the cutting stock problem (*CSP*) was one of the problems identified by Kantorovich in his 1939 paper entitled "Mathematical methods of organizing and planning production" (later published in Kantorovich (1960)) (Ben Amor and Valério de Carvalho (2005)). The *CSP* is concerned with determining the best way of cutting a set of objects into smaller items, in order to satisfy a given demand of the items and optimizing an objective function, often with a large potential of economic savings, such as the minimization of waste and the minimization of used objects. A strong characteristic of *CSP* is that the research direction has largely been motivated by taking inspiration, or directly solving, problems from industry. The literature describes a large range of problems that often include specific operational constraints and objectives, as describes the EURO Special Interest Group on Cutting and Packing (ESICUP). One of the Group main purposes is to improve communication among

individuals working in this field. The *CSP* wide variety of industrial applications includes paper, steel, wood and glass industries.

The wide range of cutting stock applications motivated Dyckhoff (1990) to introduce a typology for the cutting problem. His typology organizes the problem according to four characteristics, which are the geometric dimensions $(1, 2, 3, n > 3)$, type of assignment (i.e selection of objects and items), assortment of large objects and the characterization of the assortment of the items. Later, Wäscher et al. (2007) proposed a revised typology that provided a consistent system of problem types which allows for a complete categorization of all known cutting problems and the corresponding current literature.

The cutting process in *CSP* may be affected by various factors, particularly by the number of times one has to switch between different cutting patterns, e.g., changing the positions of the cutting knives (Wuttke and Heese (2018)) or the position of lasers. Such adjustments often interrupt production and/or may impose a setup cost every time a different cutting pattern is used, often leading to impractical applications due the use of many different cutting patterns. Therefore, it is desirable to have a cutting plan composed of fewer cutting patterns. The problem that focuses only on minimizing of the number of different cutting patterns while satisfying demands is known in the literature as the Pattern Minimization Problem (*PMP*) (Vanderbeck (2000)). In the remainder of this thesis, *PMP* will be refereed to as the single period problem with setup costs on the cutting patterns where only the minimization of different cutting patterns is considered as objective. When additional costs are considered in the objective function (trim loss or object costs), the problem will be denoted as *CSPs*. We note that it is important to have bi-criteria decision problems when setup costs are significant when compared to material costs (Yanasse and Limeira (2006)). We also remark that even the single period version of this problem is known to be $\mathcal{NP}$-hard (McDiarmid (1999)).

Considering the *CSPs* with multiple time periods (Tomat and Gradisar (2017) and Trkman and Gradisar (2007)), there is a strong relevance to the lot-sizing problem, which has been an area of very active research over the last six decades (Brahimi et al. (2017)), offering significant cost savings to the manufacturing sector by generating the least costly production plan over a planning horizon with multiple periods as well for the green manufacturing sector (e.g. when carbon emission constraints are considered (Absi et al. (2016))). The lot-sizing problem deals with key decisions such as when and how much to produce or stock, while respecting limitations such as satisfying demands on time. The lot-sizing problem can also be characterized by the number of levels in the production structure (single level or multi-level), time horizon (finite or infinite) and the presence

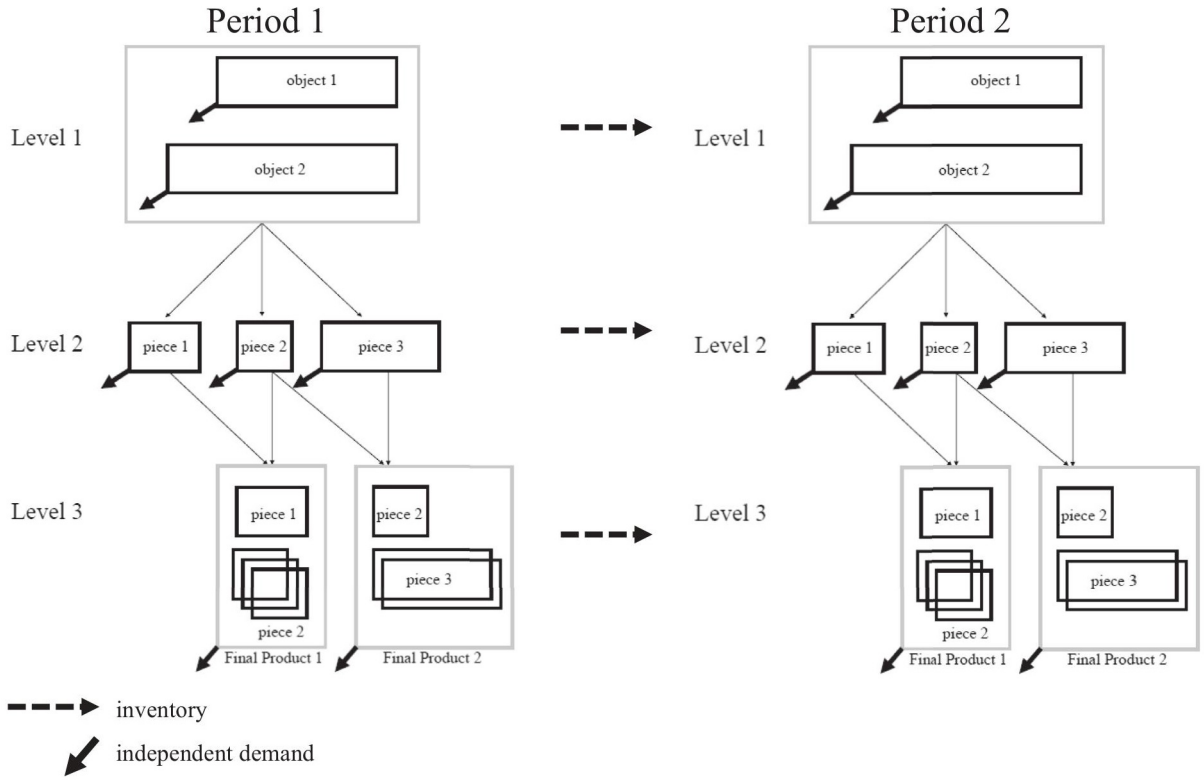
of capacity constraints. The research devoted to the topic (and solution methodologies therein) is extensive, ranging from polyhedral methods such as extended formulations and valid inequalities (Doostmohammadi and Akartunali (2018), Gruson et al. (2019), and Zhao and Zhang (2020)) to decomposition and relaxations (Akartunali et al. (2016), de Araujo et al. (2015), and Van Vyve et al. (2014)), and heuristics designed for real-world problems (Absi and van den Heuvel (2019), Fiorotto et al. (2017), and Wu et al. (2018)), as well as stochastic and robust approaches to tackle uncertainty in a broad range of settings (Alem et al. (2020), Attila et al. (2021), and Quezada et al. (2020)). An extensive analysis of lot-sizing problems can be found in the book by Pochet and Wolsey (2006).

Before considering integrated decisions during the production process, the literature has dealt with the cutting stock and the lot-sizing problems separately and sequentially. Firstly, the lot-sizing is solved and subsequently, the *CSP* is solved. However, this approach may increase the total cost, especially if the cutting process is economically relevant (Gramani et al. (2009) and Poltroniere et al. (2008)). The integration of these problems opens an interesting area for further research. Over the last decades this tendency was observed for the cutting stock and the lot-sizing. The interest in this problem often originates from direct practical applications of the integrated environments in various industries. For example, in the paper industry, large reels are manufactured or purchased and a decision about the size of the lots must be taken. Next, the large reels are cut into smaller reels that might correspond to customer requests, and a decision related to the cutting stock problem is needed (Melega et al. (2018)). Another motivation for studying these kind of problems is to explore models that capture the interdependence between both decisions in order to obtain better solutions. In general, the integrated lot-sizing and cutting stock problem consists of determining the cutting patterns and multiplicities of the corresponding cutting patterns (i.e., occurrence frequency) in each period of the planning horizon to satisfy customer demands while optimizing a given objective function, such as minimizing the costs associated with cutting pattern setup, inventory holding or objects consumed.

The integrated lot-sizing and cutting stock problems were reviewed in the extensive research of Melega et al. (2018), where a deterministic mathematical model, that considers multiple dimensions of integration and comprises several aspects found in practice, is proposed. This model is used as a framework to classify the current literature in this field. The authors essentially identify three levels of production, with the first level associated to the purchase/manufacture of object(s), the second level to the cutting process

of objects into pieces, and third level to production of final products from pieces. An example of the generalized three level integration for two periods is presented in Figure 1. In their classification scheme, the case that considers exclusively the second level, with multiple time periods in a planning horizon, the inventory of cut pieces providing the link between different periods and cutting of objects planned for each period is called the Multi-Period Cutting Stock Problem (*MPCSP*). It is also worth noticing that when more than one level is considered, with multiple time periods in a planning horizon, Melega et al. (2018) classify the problem as the Integrated Lot-sizing and Cutting Stock Problem.

Figure 1: Integration between three production levels and two time periods. Source (Melega et al. (2018))



Although integrated decision problems have gained more attention over the last decades, there are only a small number of papers dealing with setups on cutting patterns with multiple periods in the literature. Moreover, most of this literature uses the multi-period adaptation of the well-known *CSP* formulations of Gilmore and Gomory (1961, 1963) and Kantorovich (1960). The cutting stock formulations based on the model of Gilmore and Gomory (1961, 1963) are classified as pattern-based formulations. The generalized model of Melega et al. (2018) is a pattern-based model and it address most of their literature review papers considering the *MPCSP* with setups. Even though this type of formulation avoids linearity issues, the number of variables in pattern-based models is exponential in the number of items.

As we note, different types of *CSP* models derive different types of *MPCSP* models. The arc-flow based models of Alves and Valério de Carvalho (2008a), Delorme (2017), and Valério de Carvalho (1999, 2002) are a different branch for modeling the *CSP*. The model of Valério de Carvalho (1999, 2002) was firstly proposed for the *MPCSP* by Poldi and de Araujo (2016) (through without cutting pattern setup minimization). The fist *MPCSP* model considering setups on cutting pattern was proposed, to the best of our knowledge, by Ma et al. (2019) and is based on the *PMP* arc-flow of Alves and Valério de Carvalho (2008a). Another types of *CSP* model formulations are the One-cut formulation of Dyckhoff (1981) and the knapsack-based formulation of Kantorovich (1960). A non-linear knapsack-based formulation was presented by Vanderbeck (2000).

In this thesis, we consider the *MPCSP* with setup costs on cutting patterns and an one-dimensional cutting process, which will hereafter refer to it as the *MPCSPs*. This research makes important contributions in this domain. First, we present four formulations to provide a rather complete picture of alternative formulations for the *MPCSPs*. To the best of our knowledge three of these *MPCSPs* formulations are proposed here for the first time in the literature, the ones inspired by the *CSP* models of Johnston and Sadinlija (2004), Delorme and Iori (2019) and Ben Amor and Valério de Carvalho (2005). Secondly, we consider strengthening the formulations by using extended reformulations. More specifically, we use the facility location reformulation of Krarup and Bilde (1977). Although this is an effectively used method in the lot-sizing domain, its application to the *MPCSPs* is not trivial due to cutting patterns. Thirdly, we present a thorough theoretical analysis investigating the strength of various formulations given in the thesis, providing a comparative ranking with respect to lower bounds to be expected from the formulations. To complement our theoretical analysis with an understanding of performance in practice, a computational analysis is provided. Moreover, in order to solve large instances of the problem, we also propose a solution method based on the Adaptive Biased Random Key Genetic Algorithm (*ABRKGA*) presented by Chaves et al. (2018).

We remark that although the concepts of the cutting-stock and lot-sizing formulations are used, the formulations proposed are not direct adaptations from the literature of these individual problems. Considering the point of view of the *CSP*, the inclusion of setups on the single period case is already an area of extensive research (see Section 2.2). In this thesis, some of the proposed formulations are presented for the first time in the literature, even considering their simplified single period version. From the lot sizing problem point of view, since we are considering the production of cutting patterns (which contains a

set of items), the classical production variables of the Uncapacitated Lot Sizing (*ULS*) problem had to be modified and the theoretical results for the *ULS* are no longer valid. As the adaptations are not direct, an interesting problem arises, which is different from the classical *ULS* and has not been fully explored in the literature. It is important to highlight that, to the best of our knowledge, the facility location reformulation has never been applied to such a problem, and it presents high quality lower bounds.

It is worth remarking that in the *MPCSPs* addressed in this study, the setup cost only reflects the direct or indirect costs related to a setup, for example, when changeovers imply an unavoidable loss of material (Arbib and Marinelli (2007)), or when several workers are needed to perform the setup, which implies high labor cost (Kolen and Spieksma (2000)), or when a setup involves a costly craft-work (Bonnevay et al. (2016)). The considered setup costs do not include penalty costs for lost production capacity, since this should be taken into account via the introduction of setup times, which may impact time-related parameters or indicators (such as due dates, throughput, production capacity). In general, when considering practical applications, production capacity and its consequent constraints must be considered when integrating cutting stock and lot sizing problems. Since the research developed in this thesis does not consider production capacity, it is limited to some exceptional practical applications, and it is also relevant as a relaxation of several real problems, where production capacity is apparent.

The remainder chapters of this thesis are organized as follows: a literature review with models and methods related to the *MPCSPs*, as well as a discussion regarding the impact of setup costs, setup times and production capacity, is presented in Chapter 2. The formulations and their descriptions are presented in Chapter 3, in the following order: 1) a multi-period adaptation of the classical *CSP* model of Gilmore and Gomory (1961) (denoted by *AGG*), 2) an extension of the *CSP* model of Johnston and Sadinlija (2004) (denoted by *AJS*), 3) an arc-flow extension of the Alves and Valério de Carvalho (2008a) model motivated by the reflect formulation of Delorme and Iori (2019) (denoted by *ARE*), and 4) an extension of the *CSP* vehicle routing problem formulation firstly proposed by Ben Amor (1997) (denoted by *AVR*). Later, on the same chapter, we strength the formulations using the facility location reformulation and present theoretical results evaluating the strengths of the different formulations. In Chapter 4, the two heuristic procedures used in this thesis are presented, namely, the column generation technique and the *ABRKGA*. Two applications of the *ABRKGA* are proposed to solve the *MPCSPs*. Then, in Chapter 5 a discussion about the computational experiments regarding the models considering two sets of instances is presented, next, a sensitivity analysis of

the lower bounds over the object cost is presented, followed by the computational results regarding the *ABRKGA*. The chapter is then concluded with an additional experiment combining the *ABRKGA* and the column generation procedure. Finally, in Chapter 6 we make our concluding remarks and discuss some potential directions for future research.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we present a literature review related to studies that address the cutting stock problem with setup on the cutting patterns. The problem of minimizing the number of different cutting patterns in the final solution of a *CSP* has been considered since the 1970s (Haessler (1975)). Most of the papers dealing with setups on cutting patterns are an extension of the Single Stock Size Cutting Stock Problem, defined in the typology presented in Wäscher et al. (2007). In this literature review, we firstly present a discussion regarding setup costs, setup times and production capacity. Afterwards, we discuss papers that present studies for the single period cutting stock problem with setups on cutting patterns (*CSPs*), and finally, we discuss papers regarding studies for the multiple period cutting stock problem with setups on cutting patterns and, in this case, there are papers on *MPCSPs* as well as papers on the integrated lot-sizing and cutting stock problem (more than one level according to Melega et al. (2018)) that also consider setups on cutting patterns.

2.1 Setup costs, setup times and production capacity

Firstly, it is important to understand the complex trade-offs present in the objective function. In the context of the classical ULS problem, a setup cost indicates the fixed cost borne to start production, and it has a clear trade-off with inventory costs, since the larger the amount produced in a period to fulfill future demand, the smaller the incidence of fixed costs and the larger the inventory costs. In the context of this thesis, the setups refer to the action of positioning cutting knives. A clear trade-off exists between

solutions that diversify patterns (more setups) to cut fewer objects, and solutions with fewer patterns (less setups) that perhaps spend more in terms of the number of objects cut. Additionally, a clear trade-off exists between solutions that bring forward production (increasing inventory) to have better combinations of items in cutting patterns, which might decrease the total use of raw material by cutting fewer objects (Poldi, K. C. and Arenales, M. N. (2010) and Vanzela et al. (2017)). However, the trade-off between setups and inventory costs is not as clear as is for the classical ULS problem. The number of knives setups (number of setups) may only loosely, or not at all, be related to production volumes (setup run duration) and hence to inventory levels. This point is discussed in the paper of Diegel et al. (2006), where the authors attempt to avoid the problem of short setup runs in the cutting plan (which maps to avoiding small values for production amounts of each cutting pattern in each period in our problem context). According to the authors, on average, fewer number of setups mean longer setup runs, which increase the inventory levels. However, individual setup runs may not change uniformly. As the number setups decreases, some setup runs become longer, but others may remain as they are, or even become shorter. It is also worth mentioning that we are aware of only one paper that considers a multi-objective approach for a setting with multiple periods and setups on cutting patterns (Oliveira et al. (2021)), albeit without an elaboration on the points we have discussed.

Secondly, it is important to distinguish the impact of setup time from the impact of setup cost. Setups on cutting patterns have two types of major impact on production: one is the time a setup requires, which may impact on time-related parameters or indicators (due dates, throughput, production capacity, etc.); another is a direct or indirect cost derived from setup operations. Regarding production capacity, when considering single-period problems, demand is not time-indexed and hence, it is natural in this case to minimize the production time, that is, to maximize throughput. For this reason, the absence of production capacity constraints is common in the single-period case. When considering multiple cutting stock problems, production capacity and its consequent constraints must be considered to model most of the practical applications. The resource involved in the capacity constraint is time, including time spent on the number of setups and setups run duration. It is worth mentioning that the inclusion of production capacity constraints makes the problem much more difficult to solve and in general, heuristic procedures are employed.

2.2 Single period cutting stock problem with setups on cutting patterns

McDiarmid (1999) showed that *PMP* is strongly $\mathcal{NP}$-hard even for a special case where at most two items fit into a stock object, and proposed a heuristic approach to solve this special case. Regarding some important achievements from the literature, we highlight the heuristic approaches of Haessler (1988) for the one-dimensional trim-loss problem for the paper and film industries, which minimizes waste and setup costs of changing cutting patterns. Umetani et al. (2003) presented a mathematical model to the *CSPs* in which the deviation of the cut items from the demand is minimized, while the total number of different cutting patterns is considered as a constraint in the model and equal to a specific value. The proposed model is solved by an iterated local search algorithm with adaptive pattern generation, within some practical constraints on the generation of cutting patterns. Latter, Yanasse and Limeira (2006) proposed a bi-objective approach for *CSPs*. The authors proposed a hybrid procedure consisting of three phases. In the first phase, they generate cutting patterns with limited waste that fulfill the demands of at least two items when the patterns are repeatedly cut as much as possible but without overproducing any of the items; in the second phase, the reduced problem is solved and, in the third phase, a pattern-reduction technique is applied. The authors argue that the performed computational tests indicate that the proposed scheme provides alternative solutions to the pattern-reduction problem that are not overcome by other solutions obtained using procedures previously suggested in the literature.

Aloisio et al. (2011a) addressed the *PMP* for special instances, where no more than two items fit in an object in stock. They explored two formulations for the problem and derived various results concerning the existence of specific solutions. Aloisio et al. (2011b) discuss two formulations to the *PMP*, which consists of the formulation proposed by Vanderbeck (2000) and the reformulation obtained by applying the Dantzig-Wolfe decomposition. The authors show that the linear relaxation of the reformulation is stronger and as practical as the original formulation in terms of computational effort. We note that this model does not encompass a capacity constraint, i.e., it assumes the production period to be sufficiently long in order to include all the setups and to cut all the objects while limiting the objects cut to a minimum, it essentially aims at minimizing the period length.

The first branch-and-price-and-cut algorithm for *PMP* is presented in Vanderbeck (2000) which solves a compact formulation for the *CSPs*. The formulation minimizes the

number of different cutting patterns, that can be seen as an analogy to the setup on cutting patterns. The column generation approach leads to a non-linear subproblem which is decomposed into bounded knapsack problems. Alves and Valério de Carvalho (2008a) also proposed a branch-and-price-and-cut algorithm based on the model of Vanderbeck (2000). Dual-feasible functions (Alves et al. (2016)) are used for the first time to derive valid inequalities from different constraints of the model, and from linear combinations of constraints. A new arc-flow formulation is also proposed. This formulation is used to define the branching scheme of the exact method, and it allows the generation of even stronger cuts by combining the branching constraints with other constraints of the model. The computational experiments conducted on instances from the literature and random instances show that the proposed algorithm finds optimal integer solutions faster than other exact approaches.

Including the mentioned papers, a more detailed literature review of mathematical models and solution approaches to *CSPs* is presented in Henn and Wäscher (2013). Since then, more papers have been published. Cui et al. (2013) studied the two-dimensional *CSPs* where the main objective is input minimization, and pattern minimization is an auxiliary objective. The authors proposed a sequential grouping heuristic, which essentially selects the items that can be used to generate the next cutting pattern. The work of Song and Bennell (2014) focused on irregular shapes in the two-dimensional problem and employed both column generation and sequential heuristics as solution methods. The model minimizes the number of stock sheets used, while meeting the demand of items and within a constraint that limits the number of possible cutting pattern to a given value, as in Umetani et al. (2003). The work of Cui et al. (2015) used a two-phase approach to solve the one-dimensional *CSPs* also minimizing the number of stock objects. The first phase essentially generates promising cutting patterns based on a sequential grouping process and the second phase solves a mixed integer programming model. The computational results indicate the method is powerful in improving material utilization. Recently, Martin et al. (2022) proposed a pattern-based pseudo-polynomial integer linear programming (ILP) formulation for the *CSPs*. To obtain the Pareto optimal frontier, this formulation is embedded into a straightforward framework which solves the problem of minimizing the number of objects subject to a limited number of patterns in an iterative manner.

Some papers consider setup costs based on practical cases, where the quantification of the setups costs is precisely supported, i.e., setups are (or can be) priced with good precision. Arbib and Marinelli (2007) considered a real cutting process in a glass

factory, where changeovers imply an unavoidable loss of material, which can be evaluated with great accuracy since cuts are done while the glass float moves forward. In Kolen and Spieksma (2000), a practical case that arises in abrasive paper production is considered, where every setup involves a time-consuming operation that can be associated with labor cost. Bonnevay et al. (2016) studied a paper printing application, where setup involves a costly craftwork (print composition).

Evolutionary algorithms were also proposed for dealing with the setups on the cutting patterns. Golfeto et al. (2009b) introduced a symbiotic genetic algorithm for the *CSPs* minimizing trim-loss and setups on cutting patterns. The procedure works with two populations, a population of solutions (cutting plans) and a population of cutting patterns. Golfeto et al. (2009a) extended the work of Golfeto et al. (2009b) by combining it with a niche strategy to solve the problem while minimizing object consumption and setup on cutting patterns. The authors compared the results with others pattern minimization procedures such as the Sequential Heuristic Procedure of Haessler (1975), the Kombi method developed by Foerster and Wäscher (2000), the column generation based approach of Moretti and de Salles Neto (2008), the Hybrid Heuristic of Yanasse and Limeira (2006), and the Local Search Algorithm of Umetani et al. (2003). Better results are noted when higher setup costs are used. In de Araujo et al. (2014), a bi-objective approach for the *CSPs* minimizing both the number of stock objects and the number of different cutting patterns used is proposed. The authors evaluated the method using randomly generated and a practical (from a chemical fiber company in Japan (Umetani et al., 2003)) data sets. Besides the works of Foerster and Wäscher (2000), Yanasse and Limeira (2006) and Umetani et al. (2003), the authors also compared the results with the GA's of Golfeto et al. (2009a, 2009b). Their method proved to be quite efficient when considering the latter method while solving the practical set of instances. When considering the two dimensional case, the work of Mellouli et al. (2019) presented a genetic algorithm to estimate the optimal Pareto front of the *CSPs*.

2.3 Multiple period cutting stock problem with setups on cutting patterns

Studies of the *MPCSPs* in the literature are scarce. The paper by Aktin and Özdemir (2009) with an application in medicine proposed an extension of the *CSP* model of Gilmore and Gomory (1961, 1963), and involves a generalized cost function with material, setup, labor and delay costs. The authors proposed a two-stage solution method,

with the first stage estimating the total number of cutting patterns, as well as their generation, while the second stage simply solves the model. The second study consists of an application in the furniture industry (Gramani and França (2006)). There is a time limit capacity for the use of the cutting machine, where the time spent to cut one plate depends on the cutting pattern used in this plate. The authors also used the classical *CSP* model of Gilmore and Gomory (1961, 1963) with an objective to minimize trim-loss, along with inventory and setup costs. The solution approach used a shortest path reformulation of the problem. Nonås and Thorstenson (2000, 2008) studied the one-dimensional cutting stock problem in a Norwegian company that produces off-road trucks. The mathematical model consisted of a non-linear formulation with a continuous time horizon and a concave objective function, which minimizes holding costs and setup costs associated with cutting patterns. As solution methods, two global search procedures and three local search procedures were presented.

To the best of our knowledge, Ma et al. (2018) is the only paper that considered setup costs, albeit without capacity constraints, in multi-period settings. The authors proposed a mathematical model based on Gilmore and Gomory (1961, 1963) and a dynamic programming-based heuristic. The approaches were applied to real data of a problem that arises in a company that produces extra-high-voltage and high-voltage switch equipment. According to the authors, the computational results have significance for the company because it enables them to reduce total cost and enhance competitive advantages. The authors highlight that the studied problem has applications in a range of industries. In a recent paper, Ma et al. (2019) studied the *MPCSPs* with setup costs and production capacities incorporating setup duration. Two mathematical models, based on Gilmore and Gomory (1961, 1963) and the *PMP* arc-flow formulation of Alves and Valério de Carvalho (2008a), and two heuristics are presented. The first method is based on the column generation of Gilmore and Gomory. The Dantzing-Wolfe decomposition of the problem, that is similar to the one presented in Vanderbeck (2000), leads to a non-linear subproblem which is equivalent to the bounded integer knapsack problem and is solved using dynamic programming techniques. In order to obtain feasible integer solutions the authors mention three steps considering the constraint of production capacity in each period where rounding up, rounding down, or combined rounding are commonly utilized. A biased selection method is also proposed to accelerate the improvement procedure of the solutions in the previous steps. The second method is a dynamic programming-based heuristic. The general idea stems from the dynamic programming principle. However, different from the traditional dynamic programming, states are approximately aggregated.

Each pattern is explored step by step with the recursive formulation. A improvement procedure similar to that of the column generation technique is proposed. Their computational experiments highlight the efficiency of the formulation based on Gilmore and Gomory (1961, 1963) due to the fast increasing size of the arc-flow model when bigger instances are considered. Computational results show that these two heuristics are comparable and that they can obtain high-quality solutions within a reasonable time. The authors also offer sensitivity analysis and found that there is no significant difference in performance regarding the two heuristics when the parameters change. Ma et al. (2021) extended the previous work of Ma et al. (2019) by considering the production replanning problem according to demand realization which might be different from the predicted demand.

As pointed out by Melega et al. (2018), in some industrial applications, delivering the orders on time can be far more important than reducing the resulting waste and the cost of cut objects. Models that consider due dates in the formulation better describe the need of the industry in such a case (Arbib and Marinelli (2014), Braga et al. (2015), and Reinertsen and Vossen (2010)). These papers consider production capacity limits and combine the standard objective of minimizing the number of rolls used with a scheduling term penalizing the tardiness of the cutting operations. This problem is referred to in the literature as the combined Cutting Stock and Scheduling Problem. The multi-period setting also appears in these papers, but a shorter planning horizon is considered. In general, they assume that it takes exactly one unit of time to cut a stock roll.

As mentioned earlier, the multi-period cutting stock problem studied in this thesis addresses multiple periods in a finite planning horizon, the inventory of items which comprises the link between periods, and the cutting process of objects in each period. In this way, the decision-making of such processes might occur at different levels of the supply chain. For instance, the planning managers are usually responsible for the production planning of the items in order to meet the demand, whereas the machine manufacturers perform the optimization of the cuts in the cutting process. However, the literature have pointed out computational results demonstrating the benefits of an integrated approach instead of taking decisions separately (Arbib and Marinelli (2005), de Araujo et al. (2014), Gramani and França (2006), Gramani et al. (2009), Hendry et al. (1996), and Vanzela et al. (2017)). As follows, we next review those studies in which setup on cutting patterns appears in the integrated lot-sizing and cutting stock problem.

Ghidini et al. (2007) presented a mathematical model for an application in the furniture industry that addresses setup cost on the cutting patterns and production capacities

integrating setup duration, which is solved by a column generation procedure. Santos et al. (2011) approached a multi-period cutting stock problem with setup cost and time, in addition to the capacity of the cutting process, to model a real-world problem from the furniture industry. The model is solved by an optimization package considering real data from the factory and different sets of cutting patterns. In an application in the aluminum industry, Suliman (2012) presented a non-linear mathematical model, which considers setup costs for cutting patterns, as well as a capacity constraint in terms of the number of total cuts. There are also constraints to manage the inventory availability of objects with costs related to their inventories. To solve the problem, the authors present an algorithm based on a lot-sizing approach, where for each period, starting with the last one, the products and the quantities to be produced are established, as well as the cutting patterns to be used. Oliveira et al. (2021) analyzed a multi-objective integrated lot-sizing and cutting stock problem and proposed a goal programming model that takes into account six different goals representing the interests of different stakeholders in the manufacturing process. The model is solved by a column generation based heuristic. Christofoletti et al. (2021) proposed an integrated lot-sizing and three-dimensional cutting stock problem from the mattress industry. It considers setup cost and time, as well as limited production capacity. A mathematical model of mixed integer programming was proposed and solved with an optimization package. Computational results based on real data are presented. Recently, in the work of Signorini et al. (2022) a real-world production planning problem that arises in a Brazilian lattice slab factory is considered. A mathematical model based on the one-dimensional multi-period cutting stock problem with two stages is proposed. The cost function consists of the minimization of setup, production and inventory costs, subject to capacity and demand balance constraints. The model is solved by different strategies based on the column generation procedure, where two sets of subproblems are used to provide cutting patterns for each stage. Then, different rounding strategies and the relax-and-fix heuristics are applied to obtain integer solutions.

Two other papers also dealt with setups related to the cutting patterns. However, they use setup variables to estimate the setup time in the capacity constraints (Alem and Morabito (2013)) and minimum object length to use a cutting pattern (Silva et al. (2015)). Therefore, these studies do not address setup cost. Alem and Morabito (2013) proposed a two-stage stochastic mixed optimization model to represent the production planning of a small-scale furniture industry. The authors considered stochastic demand and setup times and use robust optimization tools to solve the integrated problem. Computational tests were performed using real and simulated data. Silva et al. (2015)

dealt with the problem observed in a textile factory, in which the mathematical model comprises the *CSPs*, considering upper and lower bounds on demand for each piece, with a setup constraint for each cutting pattern to guarantee a minimum length to use that cutting pattern. The model is solved by a column generation procedure. Recently, Melega et al. (2020) addressed a two-stage integrated lot-sizing, scheduling and cutting stock problem with sequence-dependent setup times and setup costs. The sequence-dependent setup appears in production stage one, where a cutting machine is used to cut large objects into smaller pieces, in which cutting patterns are generated and used to cut the pieces, and are sequenced in order to obtain a complete cutting plan for the problem. Later, in Melega et al. (2022), an integrated bin-packing and lot-sizing problem is studied. The authors considered the operation regarding the insertion or removal of the knives in the cutting process in a scenario where this procedure is sequence-dependent.

CHAPTER 3

PROBLEM DEFINITION AND MATHEMATICAL FORMULATIONS

We make the following assumptions in defining and formulating the *MPCSPs*. There is a finite planning horizon represented by a set T of periods, and a set P of item types, for which external demand is known. We consider the one-dimensional case, i.e., only one dimension is taken into account in the cutting process, and assume there is a single object type of length L, which is available in stock in an unlimited quantity. In each period $t \in T$, an item type $i \in P$ of given length l_i has to be cut from objects to meet demand d_{it}. Without lost of generality, we assume that $L \geq l_i$, L and l_i are all integral. A cutting pattern is defined as a way an object is cut to produce the demanded items. Every time a different cutting pattern is cut, a setup cost is incurred. We do not consider setup time and capacity constraints.

As the demand is known for all periods, the produced items can be brought forward to a certain period t at the cost of storing them. Without loss of generality, we also assume zero inventory for all items at the beginning of the horizon, otherwise, for each item with a positive initial inventory, this quantity can be removed from its respective demand and the value of the initial inventory can be set to zero. Therefore, the *MPCSPs* consists of determining the cutting patterns and the number of times each cutting pattern will be cut (i.e., the cutting pattern frequency) in each period of the planning horizon to satisfy customer demands, while minimizing the costs associated with cutting pattern setup, inventory of items, and objects consumed.

We also define the following notation before presenting the formulations.

Sets

T	set of periods (index t);
P	set of item types (index i);
J	set of feasible cutting patterns indices;
H_i	set of integer frequencies for item type i in object of length L.
M_t	set of cutting pattern frequencies needed to satisfy demands from period t to $\|T\|$.

For the remainder of the thesis, we consider the notation $|.|$ to denote the sets size, e.g., the set of cutting pattern integer frequencies in period t, M_t, is considered as $\{1, 2..., |M_t|\}$, where $|M_t|$ is a big enough value. In this research, we consider $|M_t| = \max_i \left\{ \sum_{\tau=t}^{|T|} d_{i\tau} \right\}$.

Input Parameters

L	length of the objects;
l_i	length of item type i;
d_{it}	demand of item type i in period t;
sc_t	setup cost of cutting patterns in period t;
uh_{it}	unit holding cost of item type i at the end of period t;
oc	material cost per object;
$\overline{a}_{ijt}$	number of items i obtained from cutting pattern j in period t.

For any period t, note that each cutting pattern j can be described by a vector $\vec{a}_j = (\overline{a}_{1jt}, \overline{a}_{2jt}, ..., \overline{a}_{|P|jt})^T$. Obviously, a cutting pattern j is valid if $\overline{a}_{1jt} l_1 + \overline{a}_{2jt} l_2 + ... + \overline{a}_{|P|jt} l_{|P|} \leq L$ holds.

We note that for pattern-based formulations, i.e. $\vec{a}_j$ is a parameter for all j, the cutting pattern frequency upper bound $|M_t|$ can be tightened by $\max_{i,j,\ \overline{a}_{ijt}>0} \left\{ \left\lceil \frac{\sum_{\tau=t}^{|T|} d_{i\tau}}{\overline{a}_{ijt}} \right\rceil \right\}$.

General Decision Variables

x_{jt}	number of objects cut according to cutting pattern j in period t;
y_{jt}	binary variable, 1 if cutting pattern j is used in period t, 0 otherwise;
s_{it}	amount of inventory of item type i at the end of period t.

The formulations proposed in this chapter are presented in the following order: Section 3.1 shows a column generation-based formulation that consists of a multi-period adaptation of the classical *CSP* model of Gilmore and Gomory (1961) (denoted by *AGG*), this formulation is the most commonly used in the literature (see Aktin and Özdemir (2009), Gramani et al. (2009), Ma et al. (2018), and Nonås and Thorstenson (2000)); Section 3.2 displays a knapsack-based formulation inspired by the *CSP* model of Johnston and Sadinlija (2004) (denoted by *AJS*); Section 3.3 presents an arc-flow-based formulation motivated by the reflect formulation of Delorme and Iori (2019) (denoted by *ARE*); finally, Section 3.4 presents the vehicle routing cutting stock problem of Ben Amor and Valério de Carvalho (2005) adapted for multiple periods (denoted by *AVR*). To the best of our knowledge, three of these formulations (*AJS*, *ARE* and *AVR*) are proposed here for the first time in the literature. In Section 3.5, we propose a facility location reformulation for these four formulations, which, to the best of our knowledge, has not been considered before for this problem setting. Then, in Section 3.6, the lower bound dominance between the proposed models is discussed. It is worth remarking that we have also implemented other formulations, such as a knapsack-based formulation motivated by Vanderbeck (2000) and an arc-flow-based formulation inspired by Alves and Valério de Carvalho (2008a). However, for the sake of keeping the exposition concise and focused, we did not include these formulations in this work, due to their not promising computational results. Though we have not used it in this thesis, it is also worth commenting that there are other single period *CSP* formulations in the literature, such as the One Cut and the Kantorovich-based formulations (Delorme and Iori (2019) and Valério de Carvalho (2002)).

3.1 Adapted Gilmore and Gomory formulation (*AGG*)

This is a multi-period extension of the *CSP* formulation of Gilmore and Gomory (1961) with setup costs on cutting patterns and is given as follows:

***AGG* model**

$$\textbf{Minimize} \sum_{t\in T}\sum_{i\in P} uh_{it}s_{it} + \sum_{t\in T}\sum_{j\in J}(sc_t y_{jt} + ocx_{jt}) \tag{3.1}$$

subject to:

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \qquad (3.2)$$

$$s_{i,t-1} + \sum_{j \in J} \overline{a}_{ijt} x_{jt} = d_{it} + s_{it} \qquad \forall i,\ \forall t \qquad (3.3)$$

$$x_{jt} \in \mathbb{Z}^+,\ y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \qquad (3.4)$$

$$s_{it} \geq 0 \qquad \forall i,\ \forall t \qquad (3.5)$$

The objective function (3.1) is the minimization of the holding costs of the items, the setup costs for the cutting patterns and the material costs of objects. Constraints (3.2) force y_{jt} to be 1 if objects are cut according to the cutting pattern j in period t, i.e., $x_{jt} > 0$. Constraints (3.3) represent the inventory balance constraints of items. Constraints (3.4) and (3.5) indicate the domains of the decision variables.

In the *AGG* formulation, when considering the cutting patterns as variables rather than parameters, non-linearity appear in the demand satisfaction constraints. The next formulation is seen as a linearized version of such non-linear formulation.

3.2 Adapted Johnston and Sadinlija formulation (*AJS*)

Johnston and Sadinlija (2004) proposed a new compact formulation for the *CSP*. Extending their ideas to the *MPCSPs* and applying it to a formulation inspired by Vanderbeck (2000), we first define binary variables r_{ijht}, which are 1 when item type i appears in cutting pattern j in period t with multiplicity h, and 0 otherwise, and then a new integer variable p_{ijht} that is equal to x_{jt} if $r_{ijht} = 1$ and 0 otherwise. We note that h is bounded by $|H_i|$. In other words, for $h \notin H_i$, where $H_i = \{1, 2, ..., |H_i|\}$, no feasible cutting patterns are generated, thus, the corresponding binary variable can be excluded from the model for all cutting patterns j. Additionally, the size of the set J needs to be defined in advance. In this thesis, $|J|$ is considered to be $|P||T|$, i.e. an upper bound for the maximum number of cutting patterns in the final solution of the problem is the number of periods times the number of item types. The formulation is then as follows:

***AJS* model**

$$\textbf{Minimize} \sum_{t \in T} \sum_{i \in P} uh_{it} s_{it} + \sum_{t \in T} \sum_{j \in J} (sc_t y_{jt} + ocx_{jt}) \qquad (3.6)$$

subject to:

$$\sum_{i \in P} \sum_{h \in H_i} h r_{ijht} l_i \leq L y_{jt} \qquad \forall j,\ \forall t \qquad (3.7)$$

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \qquad (3.8)$$

$$s_{i,t-1} + \sum_{j \in J} \sum_{h \in H_i} h p_{ijht} = d_{it} + s_{it} \qquad \forall i,\ \forall t \qquad (3.9)$$

$$p_{ijht} - |M_t| r_{ijht} \leq 0 \qquad \forall i,\ \forall j,\ \forall h,\ \forall t \qquad (3.10)$$

$$\sum_{h \in H_i} r_{ijht} \leq y_{jt} \qquad \forall i,\ \forall j,\ \forall t \qquad (3.11)$$

$$\sum_{h \in H_i} p_{ijht} \leq x_{jt} \qquad \forall i,\ \forall j,\ \forall t \qquad (3.12)$$

$$\sum_{h \in H_i} p_{ijht} \geq x_{jt} - |M_t| \left(1 - \sum_{h \in H_i} r_{ijht}\right) \qquad \forall i,\ \forall j,\ \forall t \qquad (3.13)$$

$$p_{ijht} \in \mathbb{Z}^+, \quad r_{ijht} \in \{0, 1\} \qquad \forall i,\ \forall j,\ \forall h\ \forall t \qquad (3.14)$$

$$x_{jt} \in \mathbb{Z}^+, \quad y_{jt} \in \{0, 1\} \qquad \forall j,\ \forall t \qquad (3.15)$$

$$s_{it} \geq 0 \qquad \forall i,\ \forall t \qquad (3.16)$$

The objective function (3.6) and setup constraints (3.8) are identical to previous formulation. Constraints (3.7) ensure that the items cut in an object do not exceed its length, considering the multiplicity of each item in the cutting pattern. Constraints (3.9) represent the inventory balance, where the amount produced is represented by the sum of the multiplicity of each item type i over all generated cutting patterns. Constraints (3.10) guarantee that only one r_{ijht} and its corresponding p_{ijht} can be selected for each (i, j, t) tuple because, at most, one r_{ijht} can be chosen due to constraints (3.11). Constraints (3.12) and (3.13) guarantee that if p_{ijht} is positive, then the sum of p_{ijht} regarding h must be equal to x_{jt}. Finally, constraints (3.14), (3.15) and (3.16) represent the variable domains.

3.3 Adapted Reflect formulation (*ARE*)

In Delorme and Iori (2019) a computational enhancement of the *CSP* arc-flow model, first proposed by Wolsey (1977) and later explored by Valério de Carvalho (1999, 2002), called Reflect arc-flow formulation, is proposed. In our study, the Reflect formulation is extended to the multi-period problem and a cutting pattern setup costs environment, based on the *PMP* arc-flow formulation of Alves and Valério de Car-

VALHO (2008b), is introduced.

As described in MA et al. (2019), the multi-period *PMP* arc-flow problem is formulated over acyclic directed graphs $G^{nt} = (V, A^{nt})$, $\forall n \in M_t$ and $\forall t \in T$, where $M_t = [1, 2, ..., |M_t|]$. A vertex in V corresponds to an integer position within the object, with vertex 0 representing the leftmost border of the object and L the rightmost (hence, $|V| = L + 1$). An arc (d, e) in A^{nt} represents an item of width $e - d$ placed at a position d of the leftmost border of the object. The unused portion of the objects are represented by arcs (d, e) with $e - d = 1$. Any path with tail in 0 and head in L created by arcs in set A^{nt} indicates that an object is cut n times in period t according to this path (cutting pattern).

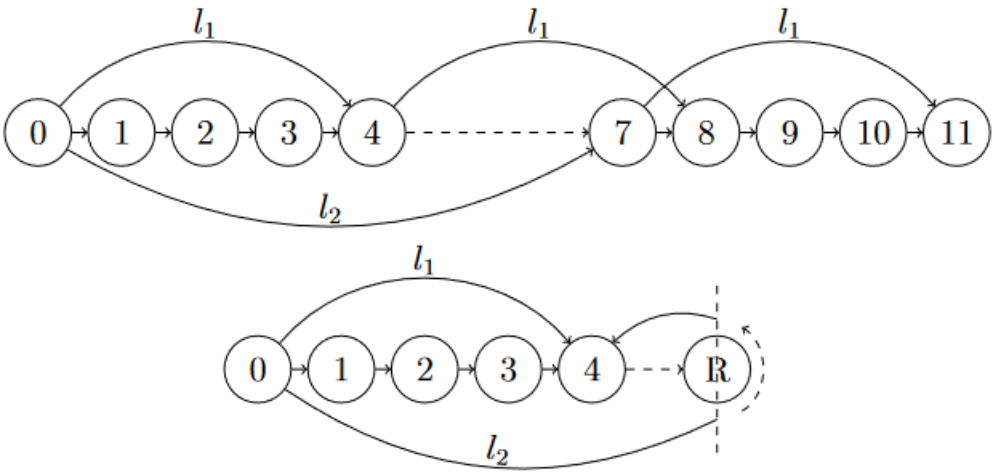
In the Reflect formulation, only half of the object capacity is considered, i.e., just $L/2$ vertices are considered. Two main features of the Reflect model are:

- it considers only vertices that are smaller than $L/2$, including 0, plus an additional node called R, which corresponds to a size of $L/2$.
- it considers the same arcs of the arc-flow formulation, but: (i) it "reflects" each arc (d, e) with $d < L/2$ and $e > L/2$ into an arc $(d, L - e)$; (ii) it removes all arcs (d, e), including loss arcs, having $d \geq L/2$; and (iii) it creates a last loss arc by connecting the rightmost node before R with R.

Intuitively, a path in the VALÉRIO DE CARVALHO (1999, 2002) formulation can be seen as a pair of colliding paths in the Reflect formulation, i.e., two paths both starting at node 0 and ending at the same node, but with only one of the two paths passing through R (the one that contains the reflected arc). DELORME and IORI (2019) proved that any feasible cutting pattern for the *CSP* can be represented by a pair of colliding paths, whose reflected arc (d, e) has $d \leq e$. Thus, reflected arcs (d, e) with $d > e$ only generate symmetric solutions and can be excluded from the model.

Figure 2 presents a standard and a reflect multigraph considering an object of length 11 ($L = 11$) and two items of lengths $l_1 = 4$ and $l_2 = 7$. For illustrative purpose, we suppose the items demands are bigger than 3. In the reflect multigraph, all the arcs before the reflected node R (5) are added to the reflect multigraph, whereas the arcs that go through the reflected node are reflected until the corresponding node. The arc with length 7 and tail in node 0 is reflected to the node 4 (11-7) while arc with length 4 and tail in node 4 is reflected to node 3 (11-8), which is removed due symmetry ($d > e$). Finally, those arcs present after the reflected node are removed from the reflect multigraph.

Figure 2: Multigraphs required for the standard arc-flow and the reflect flow, respectively. Source: author.



We now extend this idea of the reflect multigraph to the multi-period and cutting pattern setup costs settings. Maintaining the general idea of the arc-flow problem, we define multigraphs $G^{nt} = (V, A^{nt})$, $\forall n \in M_t$ and $\forall t \in T$, where the set of vertices is $V = \{0\} \cup \{r \in \mathbb{N} : 0 < r < L/2\} \cup \{L/2\}$. The set of arcs A^{nt} is then partitioned into subsets A_s^{nt} and A_r^{nt}, where A_s^{nt} represents the set of "standard" arcs and A_r^{nt} represents the set of reflected arcs, i.e., the arcs (d, e) from the arc-flow formulation (arcs from the upper graph in Figure 2) that have been reflected to arc $(d, L - e)$. Arcs in the reflect formulation (arcs from the bottom graph in Figure 2) are defined by the triple (d, e, k), where $k = s$ indicates that it is a standard arc (i.e., in A_s^{nt}), $k = r$ indicates a reflected arc (i.e., in A_r^{nt}), and (d, e) is defined as the head and tails in a fashion similar as the arcs from the standard arc-flow. We also define A_i^{nt} as the subset of arcs associated with items i in period t, i.e., $A_i^{nt} = \{(d, d + l_i, s) \in A_s^{nt},\ \forall n,\ \forall t\} \cup \{(d, L - d - l_i, r) \in A_r^{nt},\ \forall n,\ \forall t\}$. Associating an integer decision variable f_{dek}^{nt} to each arc $(d, e, k) \in A^{nt}$, $\forall n \in M_t$ and $\forall t \in T$, we can model the problem as follows:

***ARE* model**

$$\textbf{Minimize} \sum_{t \in T} \sum_{i \in P} uh_{it} s_{it} + \sum_{t \in T} \sum_{n \in M_t} \sum_{(d,e,r) \in A_r^{nt}} (sc_t + ocn) f_{der}^{nt} \tag{3.17}$$

subject to:

$$\sum_{(d,e,s)\in A_s^{nt}} f_{des}^{nt} = \sum_{(e,u,k)\in A^{nt}} f_{euk}^{nt} + \sum_{(d,e,r)\in A_r^{nt}} f_{der}^{nt} \qquad \forall e \in V\backslash\{0\},\ \forall t,\ \forall n \qquad (3.18)$$

$$\sum_{(0,e,k)\in A^{nt}} f_{0ek}^{nt} = 2 \sum_{(d,e,r)\in A_r^{nt}} f_{der}^{nt} \qquad \forall t,\ \forall n \qquad (3.19)$$

$$s_{i,t-1} + \sum_{n\in M_t} \sum_{(d,e,k)\in A_i^{nt}} n f_{dek}^{nt} = d_{it} + s_{it} \qquad \forall i,\ \forall t \qquad (3.20)$$

$$f_{dek}^{nt} \in \mathbb{Z}^+ \qquad \forall n,\ \forall t,\ \forall (d,e,k) \in A^{nt} \qquad (3.21)$$

$$s_{it} \geq 0 \qquad \forall i,\ \forall t \qquad (3.22)$$

The objective function (3.17) is the minimization of the inventory holding cost of the items, the number of reflected arcs (which is equivalent to the number of different cutting patterns), and the number of multiplicities n of each reflected arc (which is equivalent to the material cost of objects). Constraints (3.18) ensure that the amount of flow from standard arcs entering a node e is equal to the amount of flow (for both standard and reflected arcs) emanating from e plus the amount of flow from reflected arcs entering e in all periods and multiplicities. Constraints (3.19) impose boundary conditions by enforcing the amount of flow emanating from node 0 to be twice the number of different cutting patterns used. Constraints (3.20) represent the inventory balance, and constraints (3.21) and (3.22) represent the variable domains.

The arc reduction proposed by Valério de Carvalho (1999), which was later extended to the multi-period case by Poldi and de Araujo (2016), is adopted to the reflect case. Therefore, arcs with length l_i are not placed before l_k if $l_i > l_k$, $i, k \in P$ and the first loss arc is inserted in the multigraph at a distance from node 0 that is equal to the size of the smallest item. Another criterion used was the one proposed by Côté and Iori (2018), which modifies the multigraph by removing the unit-width loss arcs $(d, d+1)$ and creating longer loss arcs that connect each vertex in V to its consecutive vertex in V, for example, arcs $(4, 7)$ and $(4, R)$ in Figure 2, in the upper and bottom graphs respectively.

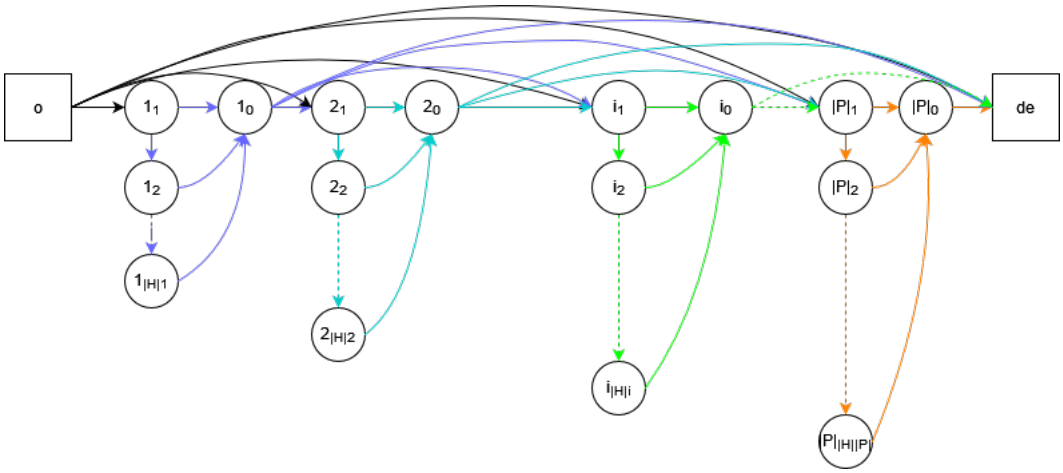
3.4 Adapted Vehicle Routing formulation (*AVR*)

Ben Amor (1997) considered feasible cutting patterns as routes in an acyclic capacitetd Vehicle Routing problem (*VRP*) network. This formulation is also discussed in Ben Amor and Valério de Carvalho (2005). In the *CSP* model as *VRP*, items are ordered in non-increasing order of length, i.e. $l_i \geq l_{i+1}$, $i \in P - \{|P|\}$. Each object corre-

spond to a vehicle of capacity L. A pair of fictive origin and destination depots represent the start (o_j) and end (de_j) nodes of a route (pattern) of vehicle $j \in J$. Each item type i (client) corresponds a set of $|H_i|$ nodes $i_1, ..., i_{|H_i|}$ and a supplementary node i_0. Two types of arcs are used: (i_h, i_{h+1}) $(h \in H_i - \{|H_i|\})$ and (i_h, i_0) $(h \in H_i)$. Any route (cutting pattern) visiting item i, begins with node i_1 and leaves at node i_0. An inter-task arc joining item types i and k, respectively, exists if $l_i + l_k \leq L$. Such an arc starts at node i_0 and ends at nodes j_1. The first, and respectively the last, arcs of a route takes the form (o_j, i_1), (i_0, de_j) $(i \in P)$. To represent a feasible cutting pattern, the routes have to respect a knapsack constraint. An item resource is used to compute the load of a vehicle (object) at each node of the network. For any $k, i \in P$, arcs of types (o_j, i_1), (k_0, i_1), and (i_h, i_{h+1}) have resource consumption l_i. Other arcs have 0 resource consumption. Let V the set of task nodes, i.e. $V = \bigcup_{i \in P}\{i_h, h \in H_i \bigcup\{0\}\}$. We define Av^{jt} as the set of arcs corresponding to a vehicle j (cutting pattern) in period t.

Figure 3 depict a network flow considering the *VRP* to construct cutting patterns. For instance, if $l_i \geq l_k$ and $h_i l_i + h_k l_k \leq L$, then a feasible cutting pattern j is generated as follows: add arc (o_j, i_1) consuming one unit of l_i and $h_i - 1$ arcs of the form (i_{1+p}, i_{2+p}), where $p \in \{1, ..., h_i - 2\}$, next an inter-task arc (i_0, k_1) is added and $h_k - 1$ arcs consuming one unit of l_k are considered. Then, arc (k_0, de_j) is considered to complete the path (route). We note that even though routes visit the vertices exactly once, no Hamiltonian cycle is formed due to the absence of feedback arcs (d_j, o_j) for each cutting pattern (route) j. We also highlight that due to the structure of the network, wherein no cycles are formed, no subtours are obtained. Therefore, no subtour elimination constraints are needed.

Figure 3: An acyclic *VRP* network for *CSP*. Source: author.



Now, to each arc (e, u) in Av^{jt} we associate a binary variable, $\hat{f}_{eu}^{jt}$, that assumes 1 if a route (cutting pattern) uses the arc (e, u) in period t and 0 otherwise, and an integer

variable, $\bar{f}^{jt}_{eu}$, to represent the number of times the cutting pattern took that specific route (via arc (e,u)) in period t only if $\hat{f}^{jt}_{eu}$ is positive. We note this model can also be seen as a multi-period extension of the VANDERBECK (2000) formulation, however, with a different linearization technique than the AJS formulation. Similarly to AJS, the set of cutting patterns (vehicles) needs to be defined in advance and therefore $|J| = |P||T|$. Hence, the *MPCSPs* formulated as a vehicle routing problem is given by:

AVR model

$$\textbf{Minimize} \sum_{t\in T}\sum_{i\in P} uh_{it}s_{it} + \sum_{t\in T}\sum_{j\in J}(sc_t y_{jt} + ocx_{jt}) \tag{3.23}$$

subject to:

$$\sum_{(o_j,u)\in Av^{jt}} \hat{f}^{jt}_{o_j u} = y_{jt} \qquad \forall j,\ \forall t \tag{3.24}$$

$$\sum_{(e,u)\in Av^{jt}} \hat{f}^{jt}_{e,u} - \sum_{(u,e)\in Av^{jt}} \hat{f}^{jt}_{ue} = 0 \qquad \forall j,\ \forall t \tag{3.25}$$

$$\sum_{(e,de_j)\in Av^{jt}} \hat{f}^{jt}_{ede_j} = y_{jt} \qquad \forall j,\ \forall t \tag{3.26}$$

$$\sum_{(o_j,u)\in Av^{jt}} \bar{f}^{jt}_{o_j u} = x_{jt} \qquad \forall j,\ \forall t \tag{3.27}$$

$$\sum_{(e,u)\in Av^{jt}} \bar{f}^{jt}_{eu} - \sum_{(u,e)\in Av^{jt}} \bar{f}^{jt}_{ue} = 0 \qquad \forall j,\ \forall t \tag{3.28}$$

$$\sum_{(e,de_j)\in Av^{jt}} \bar{f}^{jt}_{ede_j} = x_{jt} \qquad \forall j,\ \forall t \tag{3.29}$$

$$\bar{f}^{jt}_{de} \le |M_t|\hat{f}^{jt}_{de} \qquad \forall d,e \in V, \forall j,\ \forall t \tag{3.30}$$

$$x_{jt} \le |M_t| y_{jt} \qquad \forall j,\ \forall t \tag{3.31}$$

$$s_{i,t-1} + \sum_{j\in J}\left(\sum_{h\in H_i}\sum_{(u,i_h)\in Av^{jt}} \bar{f}^{jt}_{ui_h}\right) = d_{it} + s_{it} \qquad \forall i,\ \forall t \tag{3.32}$$

$$\sum_{i\in P} l_i \left(\sum_{h\in H_i}\sum_{(u,i_h)\in Av^{jt}} \hat{f}^{jt}_{ui_h}\right) \le L y_{jt} \qquad \forall j,\ \forall t \tag{3.33}$$

$$x_{jt} \in \mathbb{Z}^+,\ y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \tag{3.34}$$

$$\bar{f}^{jt}_{de} \in \mathbb{Z}^+,\ \hat{f}^{jt}_{de} \in \{0,1\} \qquad \forall j,\ \forall t,\ \forall (d,e) \in Av^{jt} \tag{3.35}$$

$$s_{it} \ge 0 \qquad \forall i,\ \forall t \tag{3.36}$$

The objective function (3.23) is the minimization of the holding costs of the items,

the setup costs for the cutting patterns and the material costs of objects. Constraints (3.24)-(3.26) enforce flow conservation for each binary variable associated to arcs of route j while requiring that y_{jt} unit of flow be shipped from o_j to de_j in period t. Constraints (3.27)-(3.30) enforce flow conservation of frequency variables for each arc of route j while requiring that x_{jt} units of flow be shipped from o_j to de_j for each period t only if the binary variables of route j ship y_{jt} unit flow from o_j to de_j in period t. Constraints (3.31) force y_{jt} to be 1 if objects are cut according to the cutting pattern j in period t, i.e., $x_{jt} > 0$. Constraints (3.32) are the demand and inventory balance constraints. Constraints (3.33) ensure that the items cut in an object do not exceed its length, considering the multiplicity of each item in the cutting pattern (knapsack constraints). Finally constraints (3.34)-(3.36) indicate the domains of the decision variables.

3.5 Facility Location Reformulation

In the lot-sizing literature, extended reformulations are often employed to overcome the poor quality of the lower bounds obtained with the linear relaxation of the original formulations. The most classical reformulations consist of redefining the variables of the original problem according to shortest path and facility location problems (see Eppen and Martin (1987) and Krarup and Bilde (1977), respectively). Such variable redefinition has been shown to describe the convex hull of the *ULS* (see Barany et al. (1984)). However, as mentioned before, since in the *MPCSPs* we are not considering individual items but cutting patterns which contain a set of items, some theoretical results proposed for the *ULS* are no longer valid, and a different and challenging problem arises that has not been fully explored in the literature. In this section, we intend to help fill this gap by presenting reformulations for the *MPCSPs*. For this, the four formulations previously presented are reformulated as a facility location problem which, though not describing the convex hull of the *MPCSPs*, obtains improved lower bounds by tightening the setup constraints. It is worth mentioning that we have not found any paper in the literature that employs the facility location reformulation for the *MPCSPs*.

An additional parameter is used in order to simplify the models based on the facility location reformulation, as follows:

$\hat{uh}_{it\tau}$: holding cost of item type i from period t to period τ, i.e., $\hat{uh}_{it\tau} = \sum_{q=t}^{\tau-1} uh_{iq}$.

Through the next subsection, we reformulate the *AGG*, *AJS*, *AVR* and *ARE* models using the facility location reformulation strategy.

3.5.1 Adapted Gilmore and Gomory Facility Location reformulation (*AGGFL*)

This is a reformulation of the *AGG* model as a facility location problem. To present this new formulation, let us observe that a cutting pattern activated in period t contributes to fulfill the demand of item type i in different periods $\tau \geq t$, then:

- $\varphi_{ijt\tau}$: portion of the frequency of cutting pattern j cut in period t to satisfy part of the demand of item type i in period τ.

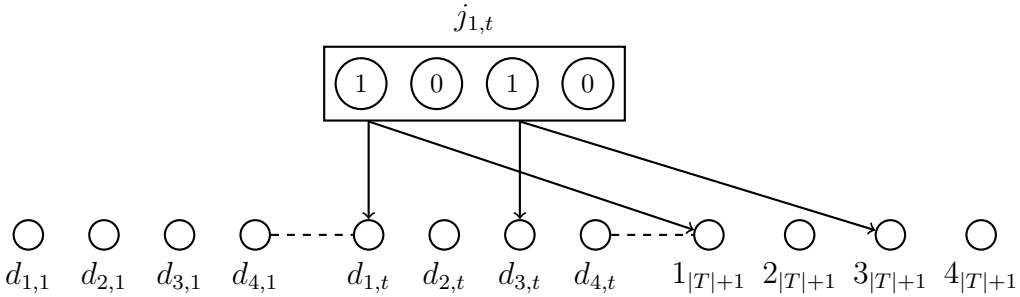
Observe that when reformulating the *MPCSPs* as a facility location problem, the relationship between the *MPCSPs* and the facility location decision variables consists of considering a facility as a period-indexed cutting pattern that must be chosen in order to perform the cutting process, and satisfy the demand of period-indexed clients. In this way, the demand (cutting) of an item type i in a future time period τ can be spread over different cutting patterns j in previous time periods, from 1 to τ, in order to be met. Mathematically, the constraints connecting the production from the cutting patterns and the production of the facilities are given by:

$$\overline{a}_{ijt} x_{jt} = \sum_{\tau=t}^{|T|+1} \overline{a}_{ijt} \varphi_{ijt\tau} \qquad \overline{a}_{ijt} > 0,\ \forall i,\ \forall j,\ \forall t. \tag{3.37}$$

Note that defining the φ variables up to period $\tau = |T|+1$ (rather than $\tau = |T|$) allows any excess production that would end up as inventory to be at the end of the horizon. In addition, the characterization of the cutting patterns ($\overline{a}_{ijt}$) in these constraints guarantee that the facility location decision variables ($\varphi_{ijt\tau}$) will have positive values for any item, only if this item belongs to the corresponding cutting pattern, i.e., $\overline{a}_{ijt} > 0$.

In order to demonstrate the relationship in constraints (3.37), an instance with an object of length 11, four item types of lengths $l_1 = 7, l_2 = 4, l_3 = 3$ and $l_4 = 2$, as well as their respective demands for $|T|$ periods, are considered. If the cutting pattern $\vec{a}_{j_1} = (1, 0, 1, 0)$ represents a facility for any given $t \in T$, the production of facility indexed with $j_{(1,t)}$ is represented as a bipartite graph, where the upper node represents the period indexed cutting pattern and the first $4|T|$ lower nodes the period indexed clients, while the remaining ones represent the production excess over the end of the horizon. Figure 4 illustrates such scheme, where each arc represents the amount of item types cut from facility $j_{(1,t)}$ to satisfy part of the demands for further periods, in other words, the arcs represent the product, $\overline{a}_{ij_1t}\varphi_{ij_1t\tau}$, for $i \in \{1, 3\}$ and $\tau \geq t$.

Figure 4: Cutting pattern production as facility production. Source: author.



Therefore, the Facility Location reformulation for the *MPCSPs* considering the *AGG* formulation is given as follows:

***AGGFL* model**

$$\textbf{Minimize} \sum_{t\in T}\sum_{i\in P}\sum_{j\in J}\sum_{\tau=t}^{|T|+1} \hat{uh}_{it\tau}\overline{a}_{ijt}\varphi_{ijt\tau} + \sum_{t\in T}\sum_{j\in J}(sc_t y_{jt} + ocx_{jt}) \tag{3.38}$$

subject to:

(3.37)

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \tag{3.39}$$

$$\varphi_{ijt\tau} \leq d_{i\tau} y_{jt} \qquad \forall i,\ \forall j,\ \forall t,\ \tau \geq t \tag{3.40}$$

$$\sum_{j\in J}\sum_{\tau=1}^{t} \overline{a}_{ijt}\varphi_{ij\tau t} = d_{it} \qquad \forall i,\ \forall t \tag{3.41}$$

$$x_{jt} \in \mathbb{Z}^+\ y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \tag{3.42}$$

$$\varphi_{ijt\tau} \in \mathbb{R}^+ \qquad \forall i,\ \forall j,\ \forall t,\ \forall \tau \geq t \tag{3.43}$$

The objective function (3.38) is the minimization of the inventory holding costs of the items, the cutting pattern setups and the used objects. The linking constraints (3.37) are added to the reformulation in order to calculate the material costs of objects in the objective function. Constraints (3.40) link the facility location and the setup variables ensuring that a setup is accounted each time a new cutting pattern is used. Constraints (3.41) guarantee the demand satisfaction and (3.42)-(3.43) define variable domains.

3.5.2 Adapted Johnston and Sadinlija Facility Location reformulation (*AJSFL*)

The *AJS* reformulation can be obtained in a similar fashion to the *AGGFL* described in Section 3.5.1. We introduce a set of φ variables with extra indexes h to obtain the link with the *AJS* production variables, i.e.:

$$p_{ijht} = \sum_{\tau=t}^{|T|+1} \varphi_{ijht\tau} \qquad \forall i,\ \forall j,\ \forall h,\ \forall t \tag{3.44}$$

The reformulation is then given by:

***AJSFL* model**

$$\textbf{Minimize} \sum_{t\in T}\sum_{i\in P}\sum_{j\in J}\sum_{\tau=t}^{|T|+1}\sum_{h\in H_i} \hat{uh}_{it\tau} h \varphi_{ijht\tau} + \sum_{t\in T}\sum_{j\in J}(sc_t y_{jt} + ocx_{jt}) \tag{3.45}$$

subject to:

(3.44)

$$\sum_{i\in P}\sum_{h\in H_i} hr_{ijht} l_i \leq L y_{jt} \qquad \forall j,\ \forall t \tag{3.46}$$

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \tag{3.47}$$

$$\varphi_{ijht\tau} \leq d_{i\tau} r_{ijht} \qquad \forall i,\ \forall j,\ \forall h,\ \forall t, \forall \tau \geq t \tag{3.48}$$

$$\sum_{j\in J}\sum_{\tau=1}^{t}\sum_{h\in H_i} h\varphi_{ijht\tau} = d_{it} \qquad \forall i,\ \forall t \tag{3.49}$$

$$\sum_{h\in H_i} r_{ijht} \leq y_{jt} \qquad \forall i,\ \forall j,\ \forall t \tag{3.50}$$

$$\sum_{h\in H_i} p_{ijht} \leq x_{jt} \qquad \forall i,\ \forall j,\ \forall t \tag{3.51}$$

$$\sum_{h\in H_i} p_{ijht} \geq x_{jt} - |M_t| \left(1 - \sum_{h\in H_i} r_{ijht}\right) \qquad \forall i,\ \forall j,\ \forall t \tag{3.52}$$

$$p_{ijht} \in \mathbb{Z}^+, \quad r_{ijht} \in \{0,1\} \qquad \forall i,\ \forall j,\ \forall h\ \forall t \tag{3.53}$$

$$\varphi_{ijht\tau} \in \mathbb{R}^+ \qquad \forall i,\ \forall j,\ \forall h,\ \forall t, \forall \tau \geq t \tag{3.54}$$

$$x_{jt} \in \mathbb{Z}^+, \quad y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \tag{3.55}$$

The objective function (3.45) is the minimization of the inventory holding costs of the items, the cutting pattern setups and the used objects. Constraints (3.44) are responsible for linking the decision variables and are added to the reformulation in order to calculate

the material costs of objects in the objective function. Constraints (3.46), (3.47), (3.50), (3.51) and (3.52) are the same as in the *AJS* model. Constraints (3.48) link the facility location and setup decision variables ensuring that a setup is accounted each time a new cutting pattern is used. Constraints (3.49) guarantee the demand satisfaction. Finally, constraints (3.53) - (3.55) define the variable domains.

3.5.3 Adapted Reflect Facility Location reformulation (*AREFL*)

Even though the cutting pattern setup is not explicitly expressed in the *ARE* model, it is still possible to obtain an extended reformulation from it. Consider the link between the facility location variables and arcs in A_i^{nt}, for a given $t \in T$, $n \in M_t$ and $i \in P$, as follows:

$$\sum_{t=\tau}^{|T|+1} \overline{\varphi}_{int\tau} = \sum_{(d,e,k)\in A_i^{nt}} n f_{dek}^{nt} \qquad \forall i,\ \forall n,\ \forall t. \tag{3.56}$$

Now, the variable $\overline{\varphi}_{int\tau}$ indexed with the integer frequency n, rather than a specific cutting pattern, represents the portion of production of each colliding path with frequency n from period t that contains arcs of item type i considered to satisfy the item type demand of period τ.

The facility location reformulation of the *ARE* model is then given as follows:

***AREFL* model**

$$\textbf{Minimize} \sum_{t\in T}\sum_{i\in P}\sum_{n\in M_t}\sum_{\tau=t}^{|T|+1} \hat{uh}_{it\tau}\overline{\varphi}_{int\tau} + \sum_{t\in T}\sum_{n\in M_t}\sum_{(d,e,r)\in A_r^{nt}} (sc_t + ocn) f_{der}^{nt} \tag{3.57}$$

subject to:

(3.56)

$$\sum_{(d,e,s)\in A_s^{nt}} f_{des}^{nt} = \sum_{(e,u,k)\in A^{nt}} f_{euk}^{nt} + \sum_{(d,e,r)\in A_r^{nt}} f_{der}^{nt} \qquad \forall e \in V\backslash\{0\},\ \forall t,\ \forall n \qquad (3.58)$$

$$\sum_{(0,e,k)\in A^{nt}} f_{0ek}^{nt} = 2 \sum_{(d,e,r)\in A_r^{nt}} f_{der}^{nt} \qquad \forall t,\ \forall n \qquad (3.59)$$

$$\overline{\varphi}_{int\tau} \leq d_{i\tau} \sum_{(d,e,k)\in A_i^{nt}} f_{dek}^{nt} \qquad \forall i,\ \forall n,\ \forall t,\ \tau \geq t \qquad (3.60)$$

$$\sum_{\tau=1}^{t} \sum_{n\in M_t} \overline{\varphi}_{in\tau t} = d_{it} \qquad \forall i,\ \forall t \qquad (3.61)$$

$$f_{dek}^{nt} \in \mathbb{Z}^+ \qquad \forall n,\ \forall t,\ \forall (d,e,k) \in A^{nt} \qquad (3.62)$$

$$\overline{\varphi}_{int\tau} \in \mathbb{R}^+ \qquad \forall i,\ \forall n,\ \forall t,\ \forall \tau \geq t \qquad (3.63)$$

The objective function (3.57) is the minimization of the inventory holding costs of the items, the cutting pattern setups and the used objects. The linking constraints (3.56) are added to the reformulation in order to calculate the material costs of objects in the objective function. Constraints (3.58) and (3.59) represent the reflect flow balance of the multigraphs. Constraints (3.60) link the facility location and setup variables ensuring that a setup is accounted each time a new cutting pattern is used for each n. Constraints (3.61) guarantee the demand satisfaction, and (3.62)-(3.63) define variable domains.

3.5.3.1 An alternative *AREFL* reformulation

The facility location variables of model (3.57)-(3.63) are $O(|T|^2|P|\sum_{n\in M_t} n)$. In this section, we present an alternative *AREFL* reformulation with facility location variable $O(|T|^2|P|)$, which obtains a weaker lower bound, but can be solved more efficiently. Consider the φ variables as the portion of the total production of item type i with respect period t, i.e.:

$$\varphi_{it\tau} = \sum_{n\in M_t} \overline{\varphi}_{int\tau} \qquad \forall i,\ \forall t,\ \forall \tau \geq t \qquad (3.64)$$

This approach mimics the variables of the facility location reformulation of the uncapacitetd lot-sizing problem, since the φ indexes only relate items and periods. The link

constraints (3.56) are replaced by:

$$\sum_{t=\tau}^{|T|+1} \varphi_{it\tau} = \sum_{n \in M_t} \sum_{(d,e,k) \in A_i^t} n f_{dek}^{nt} \qquad \forall i,\ \forall t \tag{3.65}$$

The setup constraints (3.60) are then replaced by:

$$\varphi_{it\tau} \leq d_{i\tau} \sum_{n \in M_t} \sum_{(d,e,k) \in A_i^{nt}} f_{dek}^{nt} \qquad \forall i,\ \forall t \tag{3.66}$$

In the remainder of the thesis, we will refer to this alternative formulation as *AREFL** model, in order to distinguish it from the *AREFL* model.

3.5.4 Adapted Vehicle Routing Facility Location reformulation (*AVRFL*)

The facility location reformulation of the *AVR* model can be obtained by linking the number of times a route j consumes arcs of resource l_i and the facility location variables in period t. The link is given by constraints (3.67):

$$\sum_{h \in H_i} \sum_{(u,i_h) \in Av^{jt}} \bar{f}_{ui_h}^{jt} = \sum_{\tau=t}^{|T|+1} \varphi_{ijt\tau}\ \forall i,\ \forall j,\ \forall t. \tag{3.67}$$

Hence, by adding Constraints (3.67) into the model (3.23)-(3.36) the *AVRFL* model is given by:

$$\textbf{Minimize} \sum_{i \in P} \sum_{j \in J} \sum_{t \in T} \sum_{\tau=t}^{|T|+1} \hat{uh}_{it\tau} \varphi_{ijt\tau} + \sum_{t \in T} \sum_{j \in J} (sc_t y_{jt} + ocx_{jt}) \tag{3.68}$$

subject to:

$$(3.67) \tag{3.69}$$

$$\sum_{(o_j,u)\in Av^{jt}} \hat{f}^{jt}_{o_j u} = y_{jt} \qquad \forall j,\ \forall t \tag{3.70}$$

$$\sum_{(e,u)\in Av^{jt}} \hat{f}^{jt}_{e,u} - \sum_{(u,e)\in Av^{jt}} \hat{f}^{jt}_{ue} = 0 \qquad \forall j,\ \forall t \tag{3.71}$$

$$\sum_{(e,de_j)\in Av^{jt}} \hat{f}^{jt}_{ede_j} = y_{jt} \qquad \forall j,\ \forall j \tag{3.72}$$

$$\sum_{(o_j,u)\in Av^{jt}} \bar{f}^{jt}_{o_j u} = x_{jt} \qquad \forall j,\ \forall t \tag{3.73}$$

$$\sum_{(e,u)\in Av^{jt}} \bar{f}^{jt}_{eu} - \sum_{(u,e)\in Av^{jt}} \bar{f}^{jt}_{ue} = 0 \qquad \forall j,\ \forall t \tag{3.74}$$

$$\sum_{(e,de_j)\in Av^{jt}} \bar{f}^{jt}_{ede_j} = x_{jt} \qquad \forall j,\ \forall t \tag{3.75}$$

$$x_{jt} \leq |M_t| y_{jt} \qquad \forall j,\ \forall t \tag{3.76}$$

$$\varphi_{ijt\tau} \leq d_{i\tau} \sum_{h\in H_i} \sum_{(u,i_h)\in Av^{jt}} \hat{f}^{jt}_{ui_h} \qquad \forall i,\ \forall j,\ \forall t,\ \tau \geq t \tag{3.77}$$

$$\sum_{j\in J} \sum_{\tau=1}^{t} \varphi_{ijt\tau} = d_{it} \qquad \forall i,\ \forall t \tag{3.78}$$

$$\sum_{i\in P} l_i \left(\sum_{h\in H_i} \sum_{(u,i_h)\in Av^{jt}} \hat{f}^{jt}_{ui_h} \right) \leq L y_j \qquad \forall j, \forall t \tag{3.79}$$

$$x_{jt} \in \mathbb{Z}^+,\ y_{jt} \in \{0,1\} \qquad \forall j,\ \forall t \tag{3.80}$$

$$\bar{f}^{jt}_{de} \in \mathbb{Z}^+, \hat{f}^{jt}_{de} \in \{0,1\} \qquad \forall j,\ \forall t,\ \forall (d,e) \in Av^{jt} \tag{3.81}$$

$$\varphi_{ijt\tau} \in \mathbb{R}^+ \qquad \forall i,\ \forall j,\ \forall t,\ \forall \tau \geq t \tag{3.82}$$

The objective function (3.68) is the minimization of the inventory holding costs of the items, the cutting pattern setups and the used objects. Constraints (3.70)-(3.76) represent the same as in the *AVR* model. Constraints (3.77) are the setup of the facility location variables and constraints (3.78) represent the demand satisfaction. As in the original *AVR* model, constraints (3.79) are the knapsack constraints. Finally, constraints (3.80)-(3.82) state the domain of the model variables.

Table 1 summarizes some characteristics of the *MPCSPs* formulations and reformulations discussed in this work. We mention the presence of knapsack and setup constraints and the number of binary, integer and continuous variables in each model.

Table 1: Formulations and Reformulations review.

Models	Knapsack constraints	Setup constraints	Order of Variables		
			Binary	Integer	Continuous
AGG	no	yes	$O(\|T\|\|J\|)$	$O(\|T\|\|J\|)$	–
AJS	yes	yes	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	–
ARE	no	no	—	$O(\|P\|\|T\|\sum_t \|M_t\|)$	—
AVR	yes	yes	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	–
AGGFL	no	yes	$O(\|T\|\|J\|)$	$O(\|T\|\|J\|)$	$O(\|T\|^2\|J\|\|P\|)$
AJSFL	yes	yes	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|^2\|J\|\|P\|\sum_i \|H_i\|)$
*AREFL**	no	no	—	$O(\|P\|\|T\|\sum_t \|M_t\|)$	$O(\|T\|^2\|P\|)$
AVRFL	yes	yes	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|\|J\|\|P\|\sum_i \|H_i\|)$	$O(\|T\|^2\|J\|\|P\|)$

3.6 Theoretical strengths of formulations

In this section, we intend to explore the relationship between the lower bounds of the formulations presented so far and, hence, theoretically establish the relative strengths of different formulations. Regarding the single period *CSP*, a few theoretical results can be found in the literature. Vanderbeck (2000) proved that the Dantzig-Wolfe decomposition of his formulation leads to a formulation at least as strong as the lower bound provided by the well-known Gilmore and Gomory (1961) formulation. In Valério de Carvalho (2002), the equivalence between the arc-flow formulation and the Gilmore and Gomory (1961) formulation is shown.

We use the following notation in the remainder of the thesis. Z_{LP} denotes the lower bound obtained from the linear problem (LP) relaxation of a formulation, e.g., $Z_{LP}(AGG)$ denotes the solution obtained by the AGG formulation without the integrality constraints. Likewise, X_{LP} denotes the feasible region of the LP relaxation of a formulation, e.g., $X_{LP}(AGG)$ denotes the feasible region of the AGG formulation without integrality constraints.

For the sake of clarity, we provide the following definition (from Wolsey (1998)) which will be used throughout the section.

Definition 3.1. *Given a polyhedron $Q \subset (\mathbb{R}^n \times \mathbb{R}^p)$, the projection of Q onto the subspace $\mathbf{R}^n$, denoted $proj_{(x)}(Q)$, is defined as:*

$$proj_{(x)}(Q) = \{x \in \mathbb{R}^n | (x, y) \in Q \text{ for some } y \in \mathbb{R}^p\}. \tag{3.83}$$

Proposition 3.1. $Z_{LP}(\text{AGG}) \geq Z_{LP}(\text{AVR})$, *i.e., the lower bound obtained by the* AGG *formulation is at least as strong as the one obtained by the* AVR *formulation.*

The *AVR* model is a multi-period version of the model of Ben Amor and Valério de Carvalho (2005). As proven in their paper, its Dantzig-Wolfe decomposition when

letting the knapsack and flow constraints define the subproblem leads to the *CSP* formulation of Gilmore and Gomory (1961, 1963).

Proposition 3.2. *Z_{LP}(AVR) $\geq$ Z_{LP}(AJS), i.e., the lower bound obtained by the* AVR *formulation is stronger than the one obtained by the* AJS *formulation.*

In Ben Amor and Valério de Carvalho (2005) the authors note the vehicle routing formulation for the *CSP* can be seen as a form of the *CSP* model of Kantorovich (1960). Other than for the capacity constraints adding cutting pattern setup constraints to the *CSP* model of Kantorovich (1960) leads to the *PMP* model of Vanderbeck (2000). Therefore, *AJS* and *AVR* models are both linearized versions of the Vanderbeck (2000) model. However, the linearization of the *AVR* model is accomplished through the use of flow conservation constraints resulting in a stronger model.

In fact, an indirect solution to *AJS* can be obtained by every feasible solution from *AVR* by an adequate mapping considering the flow constraints (3.24)-(3.26). For any pair j and t, if $y_{jt} > 0$, we consider, for each i, the average of the positive flow variables of the arcs in $[(o_j, i_1), (i_1, i_2), ..., (i_{|H_i|-1}, i_{|H_i|})]$. If $h_i^{'} \in H_i$ is the number of arcs with non zero flow considering item type i, we set the *AJS* variables as follows:

$$r_{ijht} = \begin{cases} \dfrac{\sum_{h \in H_i} \sum_{(u,i_h) \in Av^{jt}} \hat{f}_{u,i_h}^{jt}}{h_i^{'}}, & \text{if } h = h_i^{'}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.84}$$

Since constraints (3.24) are true, $\hat{f}_{u,i_h}^{jt} \leq y_{jt}$ for each arc $(u, i_h) \in Av^{jt}$. Therefore, constraints (3.11) will hold.

A similar thought is used considering constraints (3.27)-(3.29), let:

$$p_{ijht} = \begin{cases} \dfrac{\sum_{h \in H_i} \sum_{(u,i_h) \in Av^{jt}} \bar{f}_{u,i_h}^{jt}}{h_i^{'}}, & \text{if } h = h_i^{'}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.85}$$

then, constraints (3.12) will also hold since constraints (3.27) hold, i.e., $\bar{f}_{u,i_h}^{jt} \leq x_{jt}$ for each arc $(u, i_h) \in Av^{jt}$.

The knapsack constraints are clearly satisfied and the setup constraints (3.10) are satisfied due the setup inequalities (3.30). In order to prove that constraints (3.13) are true, we rewrite them as:

$$x_{jt} - p_{ijh_i^{'}t} + |M_t| r_{ijh_i^{'}t} \leq |M_t| \qquad \forall i,\ \forall j,\ \forall t. \tag{3.86}$$

Now, using the setup inequalities (3.30), we can establish the following upper bound for the left hand side of (3.86):

$$x_{jt} - |M_t| r_{ijh'_i t} + |M_t| r_{ijh'_i t},$$

hence, constraints (3.86) are valid since $x_{jt} \leq |M_t|$ for all j and t. Therefore, the result follows.

When comparing linear solutions from pattern-based formulations and knapsack-based formulations, a direct impact is in the frequency values of the cutting patterns. Given a linear solution from $X_{LP}(AGG)$, a linear solution on the fractional knapsack space is obtained through the following transformation:

$$\sum_{i \in P} l_i a_{ijt} y_{jt} \leq L y_{jt}, \qquad \forall j \ \forall t.$$

Then, a solution in $X_{LP}(AJS)$ is obtained by setting $r_{ijht} = y_{jt}$ if $h = a_{ijt}$ and 0 otherwise, and a solution in $X_{LP}(AVR)$ is obtained by setting a_{ijt} arc variables transporting y_{jt} units of flow, for each cut item in cutting pattern j from period t. A natural reduction in the frequency variables (x) is obtained by using the remaining space $L y_{jt} - \sum_{i \in P} l_i a_{ijt} y_{jt}$ in the stock object. Even though the "refilling" of the objects may generate new cutting patterns, the objective function value will decrease (due the smaller values of x) while the demand satisfaction constraints will remain feasible.

Proposition 3.3. *The* ARE *formulation models the* MPCSPs.

This directly follows from the fact that the reflect arc-flow formulation models the *CSP*, as proved by Delorme and Iori (2019). The authors also conclude the lower bound obtained by the reflect arc-flow, when demands are not considered as a limitation in the number of arcs, is the same as the one from the arc-flow formulation of Valério de Carvalho (1999). Therefore, it is possible to conclude that the *ARE* formulation and the *PMP* arc-flow formulation of Ma et al. (2019) provide the same lower bound. We next discuss a decomposition technique of the *ARE* formulation.

3.6.1 Dantzig-Wolfe decomposition for the flow MPCSPs formulation

In a similar fashion to Poldi and de Araujo (2016) and Valério de Carvalho (2002), we can obtain an equivalent pattern-based formulation of the *ARE* model applying the Dantzig-Wolfe decomposition to its linear relaxation while keeping the inventory

balance constraints in the master problem and letting the flow conservation constraints define the subproblem. The solutions to the subproblem are circulation flows, which include a pair of colliding paths, i.e., two paths starting at node 0 and ending at the same node, but with only one of the two paths passing through R.

The set of flow conservation constraints (constraints (3.18) and (3.19)) along with the non-negativity constraints and without the integrality requirements, define a polyhedral cone with equality constraints for each multiplicity $n \in M_t$ and period t, i.e, $P_{nt} = \{x \geq 0 : Ax = 0\}$, where A is the incidence matrix and x is a vector containing the arc-flow variables from the associated graph. From the Minkowski's Theorem (see page 96 of NEMHAUSER and WOLSEY (1988)), any point x of a non-empty polyhedron can be expressed as a convex combination of the extreme points plus a non-negative linear combination of the extreme rays of the polyhedron.

For each multiplicity n and period t, P_{nt} has just one extreme point, the origin, which corresponds to the solution with null flow. The reflect multigraph is acyclic and any circulation flow can be decomposed into a set of flows from pairs of colliding paths (see Algorithm 8 of DELORME (2017)). The circulation flow obtained by the subproblem corresponds to extremal rays of P_{nt}, so it can not be expressed as a non-negative linear combination of other circulation flows. Also, the master problem will not have a convexity constraint since the only extreme point of P_{nt} is the origin.

Now, using Algorithm 8 of DELORME (2017), the circulation flow obtained by the subproblem can be decomposed into a set of flows along colliding paths. Let Γ be the set of feasible pairs of colliding paths indices. Note that for any period or multiplicity, the set Γ will be the same. The flow along each colliding paths j, $j \in \Gamma$, is a variable of the master problem and is denoted as μ^j_{nt} for all n and t. For a given n, a column in the master problem can be defined by $(\tilde{a^j_{nt}})$, where $\tilde{a^j_{nt}} = (a^j_{1nt}, a^j_{2nt}, ..., a^j_{|P|nt})$ is the vector that defines the number of items in each period and cutting pattern multiplicity. The coefficients of these columns, a^n_{ijt}, are expressed in terms of the decision variables of the subproblem, f^{jnt}_{dek}, which correspond to the flow in the arcs (d, e, k) that take positive value when the arc is included in any of the colliding paths pair j with frequency n and is set as 0 otherwise. The relation is explicitly given by:

$$\mu^j_{nt} a^j_{int} = \sum_{(d,d+l_i) \in A^{nt}_i} f^{jnt}_{d(d+l_i)k} \qquad \forall i,\ \forall j,\ \forall n,\ \forall t \tag{3.87}$$

Note we can simply set $a^j_{int} = \overline{a}_{ijt}$ for all i, j, t and n. The first result obtained from this decomposition extend a well known results from the *CSP* literature regarding the arc-flow formulation and the formulation of Gilmore and Gomory (1961, 1963).

Proposition 3.4. *Z_{LP}(AGG) = Z_{LP}(ARE), i.e., the lower bounds obtained by the* AGG *and* ARE *formulations are equal.*

Proof. The model (3.1)-(3.5) can be derived by taking the following variable substitution into the master problem of *ARE*:

$$x_{jt} = \sum_{n \in M_t} n\mu^j_{nt} \qquad \forall j,\ \forall t \tag{3.88}$$

$$y_{jt} = \sum_{n \in M_t} \mu^j_{nt} \qquad \forall j,\ \forall t \tag{3.89}$$

□

The next result correlates the facility location reformulations of the *ARE* and *AGG* formulations.

An interesting property from the linear optimal solution of the *AGG* formulation is that the setup constraints are all satisfied at equality (else it would not be an optimal solution as setups are considered in the objective function). This property is extended for the *AGGFL* formulation and used to prove Theorem 3.1.

Theorem 3.1. *Z_{LP}(AGGFL) = Z_{LP}(AREFL), i.e., the lower bounds obtained by the* AGGFL *and* AREFL *formulations are equal.*

Proof. Using the Dantzig-Wolfe decomposition of the *AREFL* as above, we show that the optimal Z_{LP}(*M-AREFL*) lies in X_{LP}(*AGGFL*) and then that the optimal linear solution of *AGGFL* is in X_{LP}(*M-AREFL*), where *M-AREFL* is the master problem of the *AREFL* decomposition model.

$\Rightarrow$ Given the X_{LP}(*M-AREFL*) optimal solution, for n and t such that $y_{nt} > 0$, we can find i and T', $T' \subset [t, ..., |T|]$, such that Constraints (3.60) hold on equality. Therefore, for j in Γ, if $\mu^j_{nt} > 0$, we can set $\overline{\varphi}_{int\tau} = \sum_{j \in \Gamma} \varphi^j_{int\tau}$ such that:

$$\sum_{\tau=t}^{|T|+1} \varphi^j_{int\tau} = n\overline{a}_{ijt}\mu^j_{nt} \qquad \forall i,\ \forall j,\ \forall t,\ \forall n \tag{3.90}$$

$$\varphi^j_{int\tau} = d_{i\tau}\overline{a}_{ijt}\mu^j_{nt} \qquad \forall i,\ \forall n,\ \forall j,\ \forall t,\ \forall \tau \in T'. \tag{3.91}$$

Now, using Equations (3.88) and (3.89), we take the $AGGFL$ variables such that $\overline{a}_{ijt}\varphi_{ijt\tau} = \sum_{n\in M_t} \varphi^j_{int\tau}$, $\forall i,\ \forall j,\ \forall t,\ \forall \tau \in T'$. Hence, the feasibility constraints (3.37) and (3.40) are satisfied.

$\Leftarrow$ Similarly, for the linear optimal point of the $AGGFL$ model, there is at least one point i and a set T', $T' \subset [t, ..., |T|]$, such that the facility location setup constraints (constraints (3.40)) hold on equality for each positive x_{jt}. Hence, the linking constraint (3.37) referent to item i can be rewritten as:

$$x_{jt} = \sum_{\tau\in T'} d_{i\tau} y_{jt} \qquad \forall j\ \forall t. \tag{3.92}$$

Therefore, we can map the extreme point to $X_{LP}(M\text{-}AREFL)$ by taking, for each cutting pattern j in period t, $\mu^j_{nt} = y_{jt}$ and $\varphi^j_{int\tau} = a_{ij}\varphi_{ijt\tau}$ for all i, j, t and $\tau \in T'$, where $n = \sum_{\tau\in T'} d_{i\tau}$. □

Proposition 3.5. *The lower bound obtained by any of the extended facility location formulations presented in this work is at least as strong as the lower bound obtained by its respective original formulation.*

The facility location reformulation naturally tightens the convex hull of the underlying lot-sizing formulation by tightening the setup constraints (Barany et al. (1984) and Krarup and Bilde (1977)). Therefore the result follows.

Based on the previous results presented, the lower bound dominance of the formulations so far proposed can be summarized by Proposition 3.6.

Proposition 3.6. $Z_{LP}(\text{AREFL}) = Z_{LP}(\text{AGGFL}) \geq Z_{LP}(\text{ARE}) = Z_{LP}(\text{AGG}) \geq Z_{LP}(\text{AVR}) \geq Z_{LP}(\text{AJS})$.

Even though the facility location reformulations increase the lower bound of the models, one can still wonder about the relationship between $Z_{LP}(AJSFL)$ (or $Z_{LP}(AVRFL)$) and $Z_{LP}(AGG)$. In order to examine their relationship, we must understand how the facility location reformulation impacts the linear solution of the original formulation. Let's take for instance $X_{LP}(AGG)$. Considering a cutting plan containing only homogeneous cutting patterns, i.e., for each t we take $|P|$ cutting patterns such that $\overline{a}_{ijt} = 1$ if $j = i$ and 0 otherwise. Then, the extreme point $\bar{s}$ given by $s_{it} = 0$, $x_{it} = d_{it}/\overline{a}_{iit}$, $y_{it} = d_{it}/|M_t|$ $\forall i, \forall t$ does not lie in $proj_{(A,x,y,s)}(X_{LP}(AGGFL))$.

Indeed, if $\bar{s} \in proj_{(A,x,y,s)}(X_{LP}(AGGFL))$, then constraints (3.37) and (3.40) implies

that:

$$\varphi_{ijt\tau} \leq d_{i\tau} \underbrace{\frac{d_{it}}{|M_t|}}_{<1} < d_{i\tau} \qquad \forall i,\ \tau \in [t, |T|),$$

therefore, a contradiction. However, we note that if $y_{jt} = 1$ for all j and t, then, the above solution would be feasible in $X_{LP}(AGGFL)$. In fact, the new solution would be equal or greater than the previous solution (exclusively due the increment of the y variables).

In general, given the knapsack-based models of this thesis, we have two alternatives for obtaining better lower bounds. The first is to apply the Dantzig-Wolfe decomposition and obtain a pattern-based model as strong as the AGG model. The second is to apply the facility location reformulation. As discussed in the chapter, these bound-improving techniques have different impacts on the original models; the first have a direct impact on the x variable, while the second have a direct impact on the setup variables y. In the computational experiments of Chapter 5 it is shown instances where $Z_{LP}(AJSFL) > Z_{LP}(AGG)$ holds and instances where $Z_{LP}(AJSFL) < Z_{LP}(AGG)$ holds. In fact, the same is true for $AVRFL$ and therefore, we conclude that there is no dominance relationship between the AGG model and the facility location reformulation of the knapsack based models in this thesis.

CHAPTER 4

SOLUTION METHODS

Throughout this chapter, we present in more details the two heuristic procedures used in this thesis, the column generation technique and the Adapted Biased Random Key Genetic Algorithm (ABRKGA).

4.1 Column Generation

A classical approach commonly employed for solving the *CSP* pattern-based formulations is the column generation technique of Gilmore and Gomory (1961, 1963). The general idea is to initially consider only a subset of the cutting patterns of the original problem and iteratively add patterns that have the potential to improve the current value of the objective function. We next describe the procedure considering the AGG and $AGGFL$ models.

The procedure starts by defining a master problem which consists of the original model without the integrality constraints. The restricted master problem starts with the columns related to the minimal homogeneous cutting patterns, i.e., columns of type: $(0, ..., a_{ii}, ..., 0)$, where $a_{ii} = 1$, $\forall i$. The same columns are inserted for each period, $t \in T$. Then, during the pricing procedure, a knapsack problem is solved to find new cutting patterns for the restricted master problem. In the multi-period scenario, a pricing problem is solved for every period $t \in T$ and columns are included in the restricted master problem for period t until no more attractive columns are generated. The master problem, as well as the pricing subproblems are solved by an optimization package. The pricing subproblem for each period t is given by:

$$SUB(t) = \underset{a}{\textbf{Minimize}} \;\; oc - \sum_{i \in P} a_i \pi_{it} \tag{4.1}$$

subject to:

$$a_1 l_1 + a_2 l_2 + ... + a_{|P|} l_{|P|} \leq L \tag{4.2}$$

$$a_i \in \mathbb{Z}^+ \qquad \forall i, \tag{4.3}$$

where π_{it} is the dual solution associated to constraints (3.3). The objective function (4.1) is the minimization of the reduced cost from the cutting pattern. Constraints (4.2) represent the knapsack conditions and constraints (4.3) show the variable domain.

We note that because setup constraints (3.2) only exist after the cutting pattern generation, the dual variables associated to these setup constraints, π'_{jt}, for each j in period t, are not considered when solving the subproblems. Furthermore, these variables absence do not affect the column generation in terms of obtaining a valid lower bound to the problem (similar cases are studied in Melega et al. (2020) and Signorini (2021)). In fact, if we consider (4.1) as:

$$\overline{SUB}(t) = \underset{a}{\textbf{Minimize}} \;\; oc - \sum_{i \in P} a_i \pi_{it} - \pi'_{jt}, \tag{4.4}$$

since variable π'_{jt} is non-positive by duality theory, the value $-\pi'_{jt}$ will cause a non-negative increment in $\overline{SUB}(t)$, therefore $SUB(t) \leq \overline{SUB}(t)$ must hold, which assures that every cutting pattern generated by subproblem ((4.4), (4.2) and (4.3)) can be generated by subproblem (4.1)-(4.3).

Considering the *AGGFL* formulation, the procedure is the same, the restricted master is initialized with minimal homogeneous cutting patterns and the dual information from constraints (3.41) is considered to create the subproblem and generate the cutting patterns. We note that the previously argument is also used to justify the absence of the dual variables related to constraints (3.40) and (3.39), which are also associated to the cutting patterns.

Throughout the next section, basic concepts regarding evolutionary algorithms are presented. We briefly introduce the concept of genetic algorithm, followed by some of its extensions. Then, two heuristic approaches based on the Adaptive Biased Random Keys Genetic Algorithm (*ABRKGA*) of Chaves et al. (2018) are proposed.

4.2 An evolutionary algorithm for the *MPCSPs*

Evolutionary algorithms (*EA's*) perform optimization or learning tasks with the ability to evolve. Many inventions have been the result of the application of biological or natural principles to the study and design of human systems. For instance, the radar (mimics bats) and the submarine (mimics fish). The natural evolution of species could be looked at as a process of learning how to adapt to the environment and optimizing the fitness of the species (Yu and Gen (2010)).

In Yu and Gen (2010), the authors survey the viewpoint of modern genetics, i.e., "survival of the fittest" principle, in designing optimizing algorithms. According to the authors, *EA's* have three main characteristics:

- **Population-based**: a group of solutions, called a population, optimize or learn the problem in a parallel way. The population is a basic principle of the evolutionary process. The first generation of individuals is in most of cases randomly generated.
- **Fitness-oriented**: every solution in a population is called an individual. Every individual has its gene representation, called code, and performance evaluation, called fitness value. *EA's* prefer fitter individuals, which is the foundation of the optimization and convergence of the algorithms.
- **Variation-driven**: individuals will undergo a number of variation operations to mimic genetic gene changes, which is fundamental to searching the solution space.

In a general context, evolutionary algorithms are part of high-level procedures that coordinate simple heuristics to find solutions that are of better quality than those found by the simple heuristics alone, called Metaheuristic (Gendreau and Potvin (2010)).

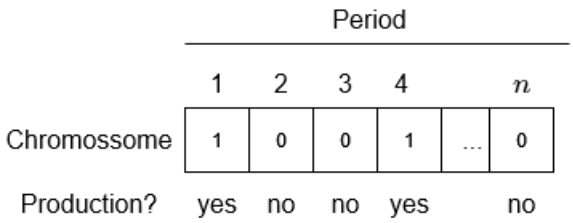
Since the 1960's, many algorithms with population-based, fitness-oriented, and variation-driven properties have been proposed. Genetic Algorithm (*GA*), as a particular case of *EA*, have been used since then. The theoretical basis for *GA's* was presented in the work of Holland (1975). The author introduced a schemata for a variety of problems such as economics, function optimization and control. We next discuss some basic concepts regarding *GA's* genetic operators.

4.2.1 Genetic Algorithms overview

On the context of *GA's*, individuals are often denoted as chromosomes, genes are the chromosome coordinates and an allele is the value taken in such coordinate. In the remainder of the chapter we also will adopt these terminologies.

Prior to implement a *GA* the user must choose a fit individual encoding in order to obtain an accurate representation of real problems. Encoding chromosomes is the first step in solving the problem and it depends entirely on the problem. The most common encoding technique is the binary encoding where each gene has its allele set as 0 or 1 representing some particular characteristic of the studied problem. In Hernandez and Suer (1999), a binary representation for the single item uncapacitetd lot-sizing problem is presented. In the authors encoding, each chromosome consists of n genes such that n is taken as the number of periods in the problem instance. The gene t, $t \in [1, n]$, indicates if an order has been placed in period t or not, Figure 5 illustrate it. The fitness function defined by the authors is the same as the usual lot-sizing objective function, which is minimize setups and holding costs.

Figure 5: Gene representation for lot sizing problem. Based on Hernandez and Suer (1999).



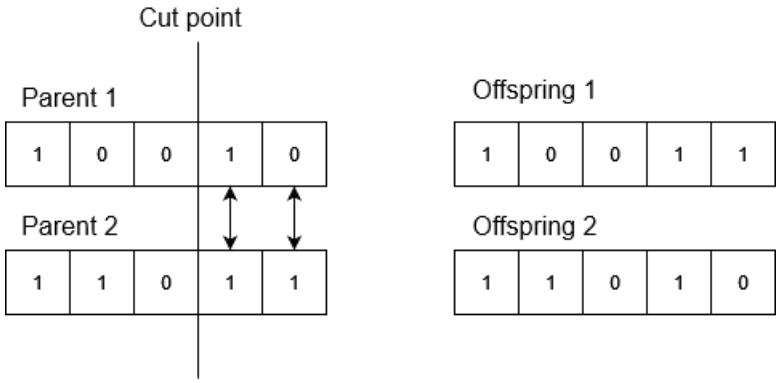
We next discuss three essential variation-driven operators for the *GA's* implementation:

- **Selection (reproduction):** in a population of p individuals, the reproduction is responsible to select the best chromosomes in order to create the next generations of candidate solutions. The most common used types are the Roulette Wheel Selection (Blickle and Thiele (1996)) and the Tournament Selection (Goldberg and Deb (1991)).

- **Crossover (recombination):** it is a genetic operator used to combine two individuals to generate a new individual. There are many types of crossover, the most used are: one-point, two-point and uniform crossovers. Figure 6 presents the one-point crossover in which a gene is randomly chosen, then the alleles on one side of the

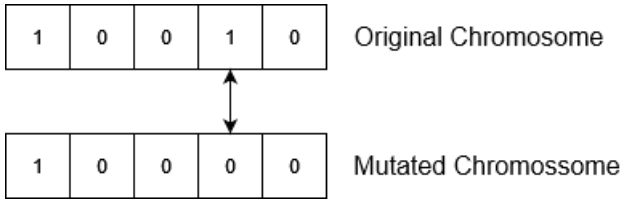
individuals are exchanged. The crossover operator may also be denoted as mating process and the resulting individuals as offspring.

Figure 6: Generating offspring. Based on Hernandez and Suer (1999).



- **Mutation:** given a probability parameter ρ_e, the mutation operator consists of a random modification in the value of one or more alleles of the individual. It is performed to every offspring, using a very low probability, for example, $\rho_e = 0.01$. Figure 7 illustrate this operator where the 5-th gene was chosen according to a given probability.

Figure 7: Mutation operator. Based on Hernandez and Suer (1999).



Then, the *GA* evolves a population over a number of iterations, called generations. The stop criteria for *GA's* may vary from time limit, maximum number of generations (max_{gen}) or solution convergence.

Even though *GA's* are widely used as metaheuristic techniques they had a slow acceptance rate in the past. This was due crossing over two feasible individuals (solutions), does not in many cases, result in a feasible solution offspring. A drawback for the use of *GA's* has been the need of specialized encoding for each problem variation in order to control the offspring feasibility. To overcome this problem, Bean (1994) introduced the concept of random keys, a method of chromosome encoding that produces feasible offspring. We next discuss some basic ideas about his method.

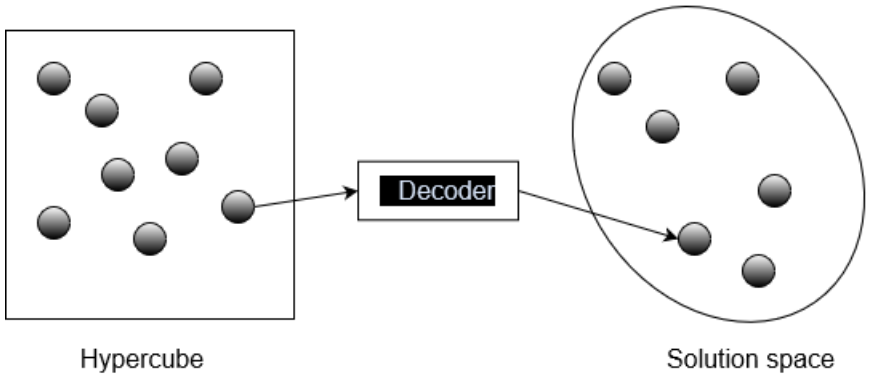
4.2.2 Randon Key Genetic Algorithm (*RKGA*)

The *RKGA* of Bean (1994) guarantees feasibility of all offspring for many sequencing and optimization problems without additional overhead. Thus, overcoming the need of specialized encoding for variations of such problems.

The random key representation encodes a solution with random numbers, typically from $(0, 1]^n$. It eliminates the offspring feasibility problems by using chromosomes that encode solutions in a soft manner. The encoding is then decoded in the objective evaluation routine in a way that avoids the feasibility problem.

The author note that the random key approach is not similar to the usual binary encoding of *GA's*. The generation of random number in the hypercubes employs a sense of randomness in conjunction with the *GA* search. No such randomness is observed in binary encoding. Figure 8 depicts the search procedure. The genetic algorithm search in the hypercube as a surrogate for the literal space. Points in the random key space are mapped to the literal space and decoded for fitness evaluation.

Figure 8: Random key process. Based on Gonçalves and Resende (2011).



In his paper, Bean (1994) presented encoders and decoders for problems such as single and multiple machine scheduling, vehicle routing, generalized traveling salesman and quadratic assignment problems. In order to show the robustness of the random key representation, we present a numerical example of the single machine scheduling problem encoder and crossover operator.

Example 4.2.1. *For the single machine scheduling,* Bean *(1994) suggests to create chromosomes where each gene is a job. For each allele we generate a uniform* $(0, 1)$ *random number. The decoding process is then accomplished by sorting the alleles and sequencing the jobs in ascending order of the sort. For instance, for a five job instance, the*

chromosome:

$$(0.36, 0.68, 0.02, 0.89, 0.53)$$

would represent the sequence $3 \rightarrow 1 \rightarrow 5 \rightarrow 2 \rightarrow 4$.

Crossover are performed over the chromosomes, the random keys. Bean *(1994) proposes the use of the parametrized uniform crossover (*Spears *and* De Jong *(1991)) instead of others often used crossover operators. After two chromosomes are randomly chosen from the population, a biased coin is toss to select which parent will contribute the allele. Assume that the head side of the coin select the allele from the first individual, a tail chooses the allele from the second individual, and that the probability of tossing a head is* 0.7 *(this value is empirically chosen). Hence, one potential crossover outcome would be:*

Coin toss		*H*	*H*	*T*	*H*	*T*		
Chromossome	1	0.36	0.68	0.02	0.89	0.53	$\equiv$	$3 \rightarrow 1 \rightarrow 5 \rightarrow 2 \rightarrow 4$
Chromossome	2	0.61	0.12	0.52	0.99	0.28	$\equiv$	$2 \rightarrow 5 \rightarrow 3 \rightarrow 1 \rightarrow 4$
Offspring		0.36	0.68	0.52	0.89	0.28	$\equiv$	$5 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 4.$

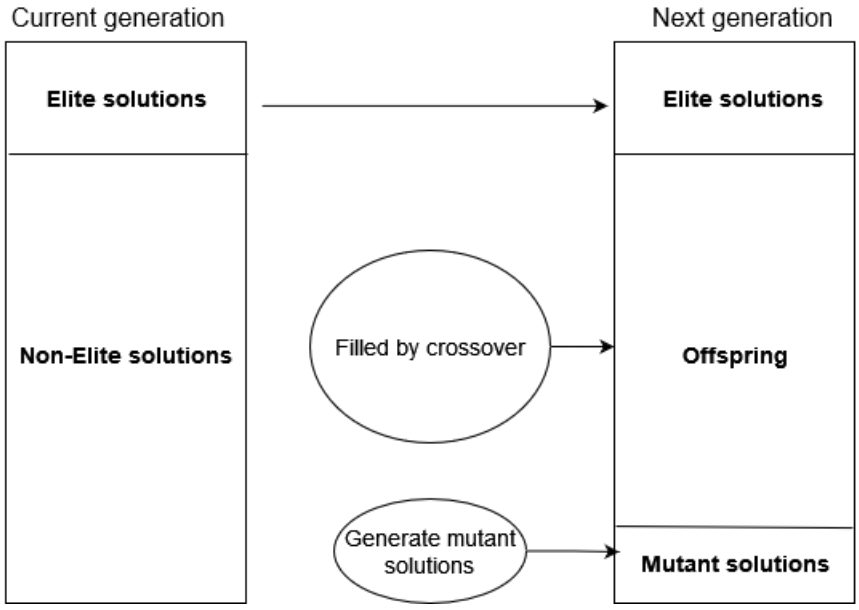
Since any sequence of numbers can be decoded into a solution sequence, all offspring are feasible solutions. The author note that through the dynamic of the GA*, jobs that should be early in the sequence evolve low numbers while jobs late in the sequence evolve large numbers. The random keys simply serve as tags which the crossover operator uses to rearrange jobs.*

The evolutionary process is then taken in place after the first p individuals are generated. The reproduction process is accomplished by copying the best individuals from one generation to the next, called a elitist strategy (Goldberg (1989)). Assume that p_e, $p_e \leq p$, is the number of elite solutions (best fitness values) kept from one generation to another. We then can divide the population into elite and non-elite solutions. A good point for using elitist rather than other non-elitist strategies is that best individuals are monotonically improved from one generation to another. The disadvantage is the population prematurely convergence in local extremes. However, it can be handled by high mutation rates.

The mutation operator is accomplished by using the concept of immigration. A number of p_m new individuals is randomly generated rather than the low probability gene by gene mutation. This process helps in prevent premature convergence of the population.

The generation transition process is presented in Figure 9. The remaining $p - p_e - p_m$ individuals for the new generation are obtained trough the crossover process described in Example 4.2.1.

Figure 9: Generational transition. Based on Bean (1994).



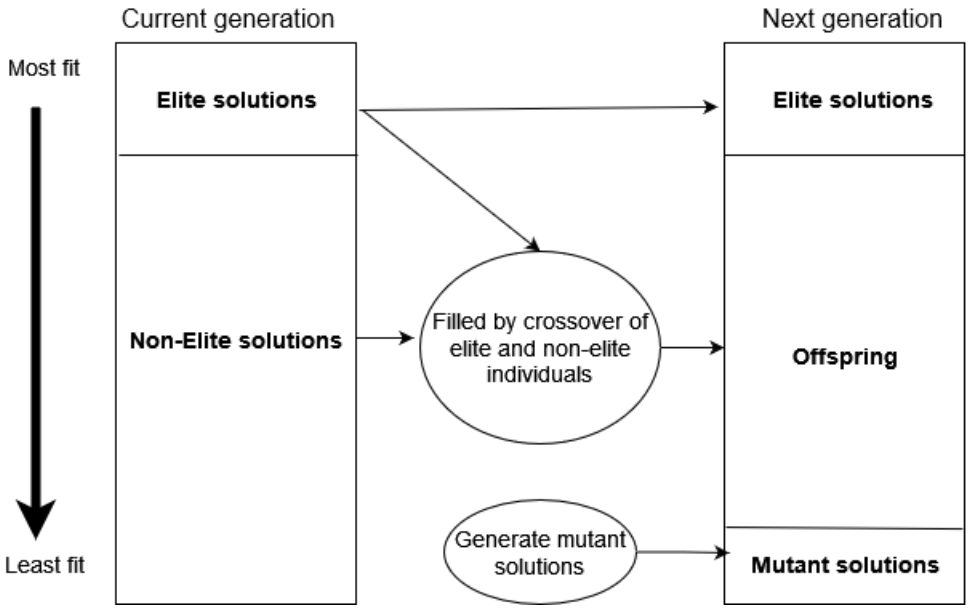
We next describe an extension of the *RKGA* proposed by Gonçalves and Resende (2011).

4.2.2.1 Biased *RKGA* (*BRKGA*)

In Gonçalves and Resende (2011), the Biased Random Key Genetic Algorithm (*BRKGA*) is proposed. The procedure differs from a *RKGA* in the way parents are selected for mating. In the *BRKGA*, during the crossover process, each child chromosome is generated combining one individual selected at random from the elite partition and one from the non-elite partition. Figure 10 illustrates the transition from generation k to generation $k + 1$.

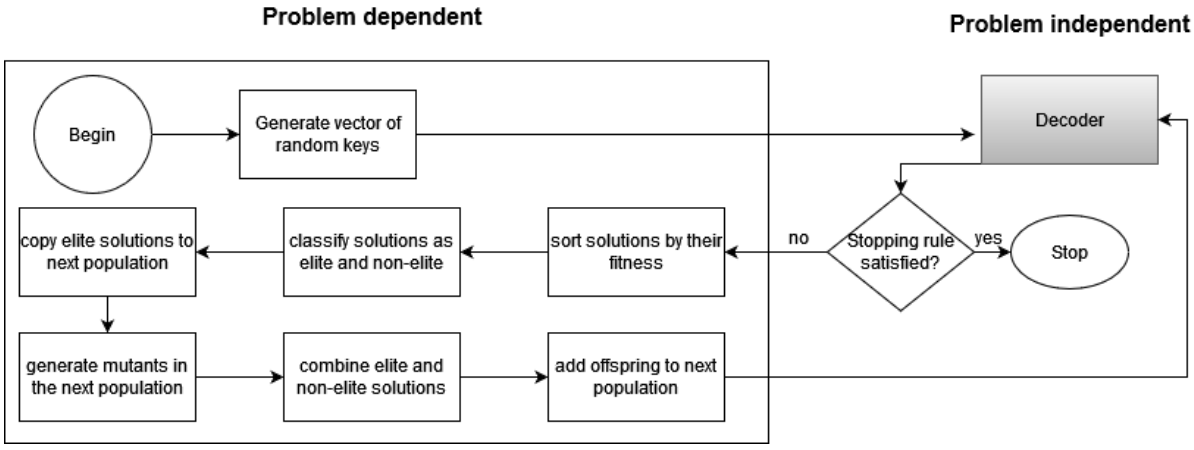
Gonçalves and Resende (2011) noted that the *RKGA* (as for *BRKGA*) are based on a general-purpose metaheuristic framework. In this framework there is a clear separation between the problem independent portion of the algorithm and the problem-dependent part. The problem independent portion has no knowledge of the problem being solved. It is limited to searching in the hypercube. The only connection to the combinatorial optimization problem being solved is the problem-dependent portion of the algorithm, where the decoder produces solutions from the vectors of random-keys and computes the fitness of these solutions. Therefore, to specify a *BRKGA* heuristic, we need only to define

Figure 10: Generational transiction in the *BRKGA*. Based on Gonçalves and Resende (2011).



how chromosomes are encoded and then decoded. This framework is depicted in Figure 11.

Figure 11: Flowchart of a *BRKGA*. Based on Gonçalves and Resende (2011).



Gonçalves and Resende (2016) performed a long survey regarding the application range of the *BRKGA* and presented a wide range of combinatorial optimization problems such as routing in *IP* networks, wavelength assignment, job-shop scheduling and constrained two-dimensional orthogonal packing problems. Encoders and decoder as well as detailed implementation and computational results are presented in the paper. The authors also recommended, trough empirically experience, the *BRKGA* parameters values as shown in Table 2.

We note that the authors did not mention an specific number of generation and the

Table 2: Recommended parameter value settings. Source: Gonçalves and Resende (2016).

Parameter	Description	Recommended value
p	size of population	$p = an$ where $1 \leq a \in \mathbb{R}$ is a constant and n is the size of the chromosome
p_e	size of elite population	$0.1p \leq p_e \leq 0.25p$
p_m	size of mutant population	$0.1p \leq p_m \leq 0.3p$
ρ_e	elite allele inheritance probability	$0.5 \leq \rho_e \leq 0.8$

value for max_{gen} varied from 500 to 8000 depending on the studied problem. Appropriate parameter choices influence on the efficiency and effectiveness of the search of specific optimization problem in an available running time. We next describe an extension of the *BRKGA* proposed by Chaves et al. (2018).

4.2.2.2 Adaptive *BRKGA* (*ABRKGA*)

Parameter setting are part of any metaheuristic. However, the values are not optimal for all problems or instances, or in terms of search time available for the runs. Thus, a careful initialization of the parameter values used in each application may be necessary (Chaves et al. (2018)).

In Eiben et al. (2007), a survey regarding the parameter setting techniques for evolutionary algorithms is conducted. The authors classified the parameter setting between off-line initialization (parameter tuning), which the parameters values can be defined before the evolution process and remain fixed throughout the run, and on-line initialization (parameter control), that the parameter values can be dynamically modified during the process. In Chaves et al. (2018), a parameter control extension of the *BRKGA*, called Adaptive Biased Random Key Genetic Algorithm (*ABRKGA*) is proposed. The technique is applied to the capacitated centered clustering problem.

In the *ABRKGA*, two new parameters, α and β, are introduced aside the usual *RKGA* and *BRKGA* parameters. Then, in the sense of the on-line initialization, the parameters p, p_e, p_m and α are updated with deterministic rules while β and ρ_e are self-adaptive. The user needs to set only a new positive parameter λ, $\lambda \leq 1$, based in the running time available. The parameters ρ_e and β are computed via two extra random keys inserted in genes $n+1$ and $n+2$ of each individual χ, $\chi \in (0,1]^n$, in the population.

The parameter λ is also used to calculate max_{gen} via Expression (4.5):

$$max_{gen} = \lambda^{(\log_\lambda(p_{min}) - p_{max})}, \tag{4.5}$$

where p_{max} and p_{min} denote the maximum and minimum number of individuals in a population, respectively. The *ABRKGA* iterations start with $p = p_{max}$. The generation transition of the *ABRKGA* is given as follows:

- In each generation the population size is decreased by $p^{k+1} = p^k\lambda$, where p^k is the population size for generation $k \in [1, max_{gen}]$;
- the size of the elite and mutant partition are given by $p_e = max\{3, \kappa_e p\}$ and $p_m = max\{1, \kappa_m p\}$, respectively, such that:

$$\kappa_e = 0.1 + \frac{g_k}{max_{gen}}(0.25 - 0.1), \tag{4.6}$$

and

$$\kappa_m = 0.05 + \left(1 - \frac{g_k}{max_{gen}}\right)(0.25 - 0.05) \tag{4.7}$$

where g_k represents the current generation in numeric value. Thus, the maximum elite partition has few individuals in initial generations when the average quality is poor, and this number tends to increase during future generations. On the other hand, the number of mutants is decreasing along the search process.

- Besides the parameter control, the technique differs from the *BRKGA* in the way the top partitions are constructed in each generation. After the chromosomes of one generation are sorted according to their fitness, the population is partitioned into two groups, where best individuals are kept in a Restricted Chromosome List (RCL). We note the size of the RCL is limited by p_e, i.e. $|RCL| \leq p_e$. The RCL is formed by individuals whose fitness is greater than a threshold value (f_e) defined by:

$$f_e = f_{min} + \alpha(f_{max} - f_{min}), \tag{4.8}$$

where f_{min} and f_{max} are, respectively, the worst and best individuals fitness in current population, and α is computed by:

$$\alpha = 0.1 + \left(1 - \frac{g_k}{max_{gen}}\right)(0.3 - 0.1). \tag{4.9}$$

The parameter α interval is defined due to the poor quality solutions in the initial generations. It starts at 0.3 and decreases along the generations due the flatness tendency of population after a certain number of generations.

- A perturbation scheme is also proposed by the authors to allow the individuals es-

cape from local extremes. If two individuals have the same fitness, the perturbation scheme uses a number of random swaps in a random-keys vector, and the perturbation strength is defined by the self-adaptive parameter β that is the probability of swap two random keys. This parameters is calculated for each individual using the allele of the $(n+2)$-th gene, $\chi(n+2)$, as follows:

$$\beta = 0.001 + \chi(n+2)(0.1 - 0.001). \tag{4.10}$$

- After the RCL partition is obtained, p_m individuals are randomly generated to make part of the mutation partition of the population.

- The next $p - |RCL| - p_m$ individuals to obtain the whole population are obtained though the parametrized uniform crossover process. Individuals from the RCL and non-RCL set are randomly chosen to perform the crossover (mating) process such that the self-adaptive parameter ρ_e is obtained using the $(n+1)$-th allele of the non-RCL parent as follows:

$$\rho_e = 0.65 + \chi(n+1)(0.8 - 0.65). \tag{4.11}$$

As mentioned, the number of offspring is adapted to generate the new population size according to the parameter λ. Chaves et al. (2018) proposed three values, $0.999, 0.998$ and 0.997, each one according to the user intent to adopt a reasonable, moderate or high running time, respectively. The authors also mention that a restart procedure is performed when population size is less than p_{min} individuals. In this scenario, only RCL individuals are maintained in current population and p_{max} - $|RCL|$ new random individuals are generated. The remaining parameters intervals are summarized in Table 3.

Table 3: Value settings according to the *ABRKGA* parameters control. Source: Chaves et al. (2018).

Parameter	Description	Recommended value
p	size of population	$p \in [100, 1000]$
p_e	size of elite population	$p_e = max\{3, \kappa_e p\}$ where $\kappa_e \in [0.1, 0.25]$
p_m	size of mutant population	$p_m = max\{1, \kappa_m p\}$ where $\kappa_m \in [0.05, 0.2]$
ρ_e	elite allele inheritance probability	$0.65 \leq \rho_e \leq 0.8$

We note that the *ABRKGA* code is available in $c++$ language in Professor's Chaves webpage (https://sites.google.com/site/antoniochaves/publications/software?authuser=0). Next, two approaches to use the *ABRKGA* to solve the *MPCSPs* are discussed.

4.2.3 An ABRKGA for the MPCSPs

The individual representation and the decoder in this thesis are based on the papers of Gonçalves and Resende (2013) and Gonçalves and Wäscher (2020). Both papers proposed a decoder where the items are assigned one at a time in a stock object ($2D$ and $3D$ objects). For this, the authors encoded the $BRKGA$ individuals as a "soft" solution of the problem where the feasible solutions are indirectly obtained trough the chromosome random keys using a decoder procedure. In Gonçalves and Resende (2013) a multiple Phase chromosome decoding for the two and three dimensional bin packing problem is proposed. Given a set of boxes, B, to be packed, the chromosome size is set as $2|B|$ where the first part encode the boxes packing sequence and the second part encode the boxes orientations. The decoder process proposed by Gonçalves and Resende (2013) is composed as follows:

1. The first Phase decodes part of the chromosome into the sequence in which the boxes are packed.
2. The second Phase decodes part of the chromosome into the vector of box orientations.
3. The third Phase makes use of Phases 1 and 2 and construct the cutting patterns.

In Gonçalves and Wäscher (2020), a similar encoding and decoding process is proposed for the two-dimensional CSP with defects.

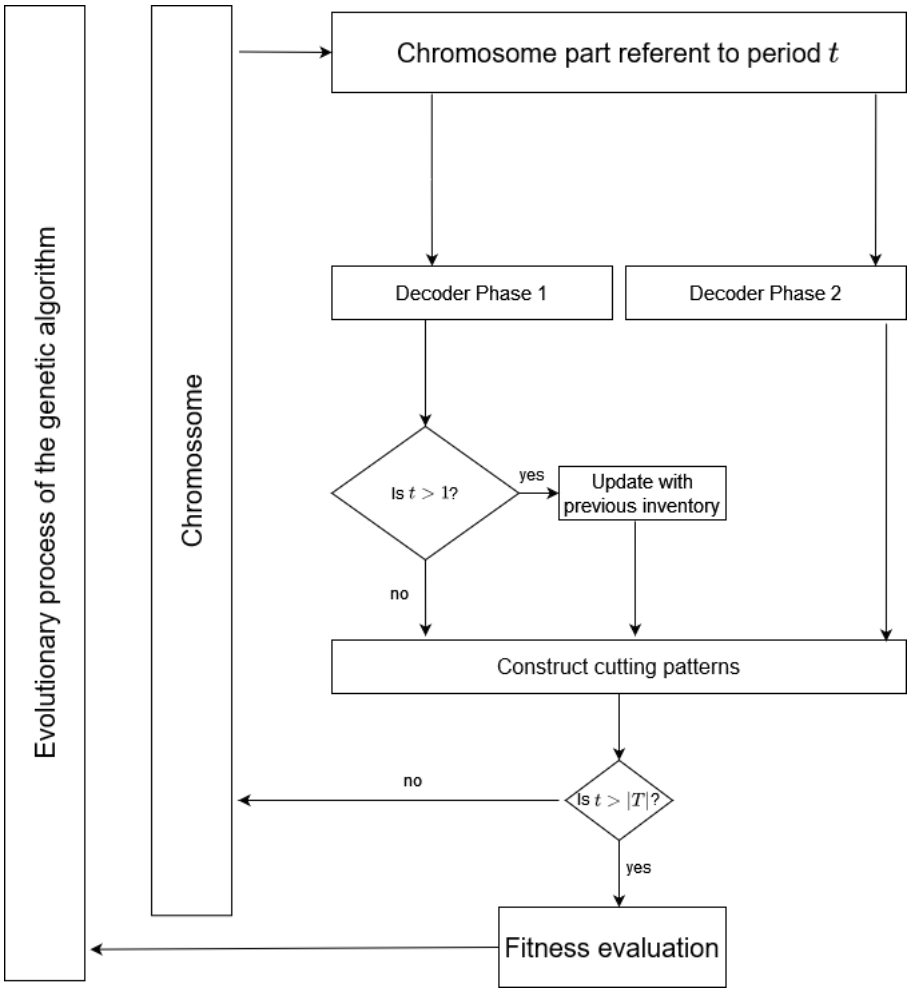
4.2.3.1 General Overview

For the $MPCSPs$, the individuals are encoded as a vector of production and parameters for the cutting pattern construction. For each period t, the decoder process consists as follows:

1. The first Phase determines, for each item, the production amount that must produce in the period. If $t > 1$, the inventory from period $t - 1$ is subtracted from this suggested production for each item.
2. The second Phase determines the rules obtained by the encoded parameters to construct the cutting patterns.
3. Next, the information from Phase 2 is used to construct the cutting patterns and calculate their frequency in order to satisfy the demand from Phase 1.

At the end of the planning horizon a feasible solution for the problem is generated and its fitness is given as the upper bound obtained by the solution in the objective function of the problem. A general framework of the decoder process is depicted in Figure 12. Two decoder processes with different Phases 1 and 2 are proposed next.

Figure 12: Decoder framework. Source: author.



4.2.3.2 Decoder 1 (D_1)

For this decoder, the chromosome is encoded as a vector of item productions, X, items occurrences, W, and items assignments, RK. The first Phase will act only on vector X while the second Phase maps on vectors W and RK. The length of W and RK are the same and equal to $|P||T|$, where $|P|$ genes are used for each vector in each period, therefore, a total of $2|P||T|$ random keys are required for Phase 2. Since X represents item production and the only possible production in period $|T|$ is the actual item demand, only $|P|(|T|-1)$ random keys will be required for Phase 1. Hence, the length of an individual, χ, when considering D_1, is $|P|(|T|-1)+2|P||T|$.

Example 5.3.1. *In order to understand the decoder in practice, we consider a small size instance with the following characteristics.* $L = 10$, $|T| = 3$ *and* $|P| = 3$. *The items length and demands are shown in Table 4.*

Table 4: Instance specifications.

Items	Demands		
	$t=1$	$t=2$	$t=3$
$l_1 = 2$	6	3	10
$l_2 = 3$	5	5	7
$l_3 = 7$	1	6	7

We also consider the following random keys vector:

$$\chi = \Big(\overbrace{.5,\ .87,\ .02,}^{t=1}\ \overbrace{.9,\ .19,\ .9,}^{t=2} \underbrace{.3,\ .28,\ .05,}_{t=1}\ \overbrace{.07,\ .3,\ .51,}^{t=2}\ \overbrace{.2,\ .63,\ .3,}^{t=3} \underbrace{.14,\ .28,\ .05,}_{t=1}\ \underbrace{.7,\ .4,\ .51,}_{t=2}\ \underbrace{.62,\ .33,\ .3}_{t=3} \Big). \quad (4.12)$$

The random keys in the first line of the individual χ are encoded for Phase 1, i.e. related to vector X, while the second and third lines are encoded for Phase 2, .i.e. used to obtain vectors W and RK, respectively. Except for line 1 with 6 random keys, the remaining line consists of 9 random keys with 3 random keys per period in the order of l_1, l_2 and l_3. △

In order to generically identify the genes position in each encoded vector, we make use of notation $it + (k-1)|P||T|$, where $k = 1$ refers to vector X, $k = 2$ refers to vector W, and $k = 3$ refers to vector RK, e.g., for $k = 2$, $\chi(it + |P||T|)$ represents a random key in vector W from a gene related to item i in period t. We next describe how to decode the random keys vector.

- In Phase 1, the vector X is obtained as follows: for each $t \in [1, 2, ..., |T| - 1]$ and $i \in P$, $X_t(i)$ is obtained as the sum of demands of item type i from period t up to period τ_{it}, such that:

$$\tau_{it} = t + fint(\chi(it)(|T| - t)), \tag{4.13}$$

where $fint$ is the function that rounds to the nearest integer. Mathematically, we have:

$$X_t(i) = \sum_{\tau=t}^{\tau_{it}} d_{i\tau}. \tag{4.14}$$

We note that $X_{|T|}(i) = d_{i|T|}\ \forall i$.

Considering the Example 5.3.1, the calculations to obtain X are as follows:

1. multiply the coordinates of $(2, 2, 2, 1, 1, 1)$ (vector formed by $|T| - t$ for each item and $t \in \{1, ..., |T| - 1\}$) with the coordinates of the first line of χ;
2. the obtained vector is given by $(1, 1.74, 0.04, 0.9, 0.19, 0.9)$;
3. applying $fint$ to the above vector, we obtain $(1, 2, 0, 1, 0, 1)$;
4. the production is then formed by the items demands from periods $(\overbrace{1+1,1+2,1+0}^{t=1}, \overbrace{2+1,2+0,2+1}^{t=2})$.

Therefore, the Phase 1 decoded from χ will give the following vector of productions for each item in each period: $X = (\overbrace{9,17,1}^{t=1}, \overbrace{13,5,13}^{t=2})$.

- In Phase 2, the vector W represents the number of occurrences of the items while RK represents an order in which items are assigned to the stock object.

1. for each i and t, the item occurrence, $W_t(i)$, is obtained using a simple rounding technique based on the random key and the maximum occurrence item type i may appear in a cutting pattern. The formula is given by:

$$W_t(i) = \lceil \lfloor L/l_i \rfloor \chi(it + |P||T|) \rceil. \tag{4.15}$$

Considering the Example 5.3.1, the calculations to obtain W_t are as follows:

(a) the vector of maximum item occurrence for each item in each period is $(5, 3, 1, 5, 3, 1, 5, 3, 1)$;

(b) multiplying the coordinates from the above vectors with the coordinates from the second line of χ, we obtain $(\underbrace{.1.5, \quad .0.84, \quad .05}_{t=1}, \underbrace{.35, \quad 0.9, \quad .51}_{t=2}, \underbrace{1, \quad 1.89, \quad .3}_{t=3})$.

Hence, applying the ceil function on the above vector, the first information decoded from Phase 2 is the vector of item occurrences: $W = (\overbrace{2,1,1}^{t=1}, \overbrace{1,1,1}^{t=2}, \overbrace{1,2,1}^{t=3})$.

2. For each period t, the vector RK_t will have each coordinate associated to an item in the following way:

$$RK_t = (\chi(1t + 2|P||T|), \chi(2t + 2|P||T|), ..., \chi(|P|t + 2|P||T|)). \tag{4.16}$$

The assignment order is obtained by sorting in decreasing order the random keys of RK_t. The resulting sequence $\{i_1, i_2, ..., i_{|P|}\}$, $i_k \in P$ for each $k \in \{1, ..., |P|\}$, displays an assignment order. For the Example 5.3.1, the assignment order obtained from χ is given by: $RK = (\overbrace{l_2, l_1, l_3}^{t=1}, \overbrace{l_1, l_3, l_2}^{t=2}, \overbrace{l_1, l_2, l_3}^{t=3})$.

- **Cutting pattern construction:** in order to obtain a feasible solution for the *MPCSPs*, we construct, for each period t, a set of cutting patterns, J_t, considering the decoded information from Phases 1 and 2. An auxiliary function, $\sigma_t : P \to \{0, 1\}$, indicates that item type i will be used in a cutting pattern in period t if $\sigma_t(i) = 1$ and 0 otherwise. We suppose at first that $\sigma_t(i) = 1$ $\forall i \in P$. Then, for each t, the information from vector W_t (Equation (4.15)) and vector RK_t (Equation (4.16)) derives the vector of lengths, LV_t, which has size $|P|$ and is composed by the items space to be occupied in an object of size L, i.e., $LV_t = (l_{i_1}W_t(i_1), l_{i_2}W_t(i_2), ..., l_{i_{|P|}}W_t(i_{|P|}))$ where each item type i is assigned in a specific order (from RK_t) with a specific occurrence (from W_t). Then, starting from $l_{i_1}W_t(i_1)$ the lengths are sum up until an item type assignment exceeds the capacity L, e.g., if $l_{i_1}W_t(i_1) \leq L$ and $l_{i_1}W_t(i_1) + l_{i_2}W_t(i_2) > L$, then item type i_2 will not be assigned to the cutting pattern and a pre-processing over the frequency $W_t(i_2)$ is conducted to find a feasible amount to fit in the object. If a smaller frequency for i_2 is found, then its random key in the second part is updated accordingly and the algorithm starts a new cutting pattern, otherwise it checks the next item type in the sequence using the frequency pre-processing technique if necessary.

- **Cutting pattern frequency:** after a cutting pattern j is constructed, the items cut are classified as "used", i.e., $\sigma_t(i) = 0$ if $a_{ijt} > 0$, then the cutting pattern frequency,

x_{jt}, is calculated using Phase 1 information such that:

$$x_{jt} = \max_{i \in P} \left\{ \left\lceil \frac{X_t(i)}{a_{ijt}} \right\rceil, a_{ijt} > 0 \right\}. \tag{4.17}$$

Then the process restarts considering the non used items until $\sigma_t(i) = 0 \; \forall i \in P$. The process for a specific period is outlined in Algorithm 1. After the construction in period t, $t < |T|$, stops, the inventory is calculated and the parameters from Phase 1 in the next period, $X_{t+1}(i)$, are updated accordingly. We note that if $X_{t+1}(i) \leq 0$ for some i, then $\sigma_{t+1}(i)$ is set as 0.

Algorithm 1: Cutting pattern generation for period t.

Vector Input: $W, LV_t, (\sigma_t(i_1), ..., \sigma_t(i_{|P|}))$ and $a \in \mathbb{R}^{|P|}$.
Int Input: $L^{'}$.
Output: J_t.

$J_t \leftarrow \emptyset$, $a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$;
while $(\exists i \in P | \sigma_t(i) = 1)$
  for $p = 1$ *to* $|P|$
    if $(\sigma_t(i_p) = 1$ *and* $L^{'} + LV_t(p) \leq L)$
      $\sigma_t(i_p) = 0$;
      $L^{'} \leftarrow L^{'} + LV_t(p)$;
      $a(i_p) \leftarrow W_{i_p t}$;
    if $(\sigma_t(i_p) = 1$ *and* $L^{'} + LV_t(p) > L)$
      find maximum $k \in \mathbb{Z}^+$ such that $k l_{i_p} \leq L - L^{'}$;
      if $k > 0$
        $\sigma_t(i_p) = 0$;
        $\chi(i_p t + |P||T|) = k / \lfloor L / l_i \rfloor$;
        $LV_t(p) \leftarrow l_{i_p} k$;
        $L^{'} \leftarrow L^{'} + l_{i_p} k$;
        $a(i_p) \leftarrow k$;
  $J_t \leftarrow J_t \cup \{a\}$;
  $a \leftarrow \mathbf{0}$;

To illustrate the construction process, we consider the instance presented in Table 4.

Example 4.2.2. *(***Continuation***) Starting from period* 1*, the decoded information for Phases 1 and 2 is shown in Table* 5*.*

Table 5: D_1 in practice for period 1.

Item	X_1	W_1	RK_1
l_1	$6+3$	2	2^{nd}
l_2	$5+12$	1	1^{nd}
l_3	$1+0$	1	3^{nd}

The cutting pattern construction Phase for period 1 *proceed as follows:*

- *The length vector* (LV_1) *derived from Phase 2 is* $(l_2, 2l_1, l_3)$*. This vector presents the order in which items length frequencies are meant to be used. Since* $l_2 + 2l_1 = 7$ *and* $L - 7 < kl_3$ *for any* $k \in \mathbb{N}$*, the third and last item type can not have it frequency random key manipulated to fit into the cutting pattern. Therefore, the following cutting patterns are generated:*

$$\vec{a}_1 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \qquad \vec{a}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.18}$$

The next step of the decoder is then applied to obtain the cutting pattern frequencies given by Equation (4.17)*. Hence:*

$$x_{11} = max\left\{\left\lceil\frac{9}{2}\right\rceil, \left\lceil\frac{17}{1}\right\rceil\right\} = 17, \qquad x_{21} = max\left\{\left\lceil\frac{1}{1}\right\rceil\right\} = 1. \tag{4.19}$$

Therefore, the following inventory is generated: $s_{11} = 34 - 6 = 29$*,* $s_{21} = 17 - 5 = 12$*, and* $s_{31} = 1 - 1 = 0$*.*

Now, the decoded information for period 2 *is given in Table* 6*.*

Table 6: D_1 in practice for period 2.

Item	$X_2 - s_1$	W_2	RK_2
l_1	$(3+10)-29$	1	1^{nd}
l_2	$5-12$	1	3^{nd}
l_3	$(6+7)-0$	1	2^{nd}

We next apply the cutting pattern construction process with Phase 1 and 2 information from Table 6*.*

- *When $t = 2$, the length vector (LV_2) is given as (l_1, l_3, l_2). Note that only item type 3 has a positive production, i.e., $\sigma_2(1) = \sigma_2(2) = 0$, therefore, only the following cutting pattern will be generated:*

$$\vec{a}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.20}$$

The frequency for $\vec{a}_3$ is given as:

$$x_{32} = max\left\{\left\lceil \frac{13}{1} \right\rceil\right\} = 13. \tag{4.21}$$

Then, the inventory levels are updated: $s_{12} = 29 - 3 = 26$, $s_{22} = 12 - 5 = 7$, and $s_{32} = 13 - 6 = 7$.

The decoded information regarding period 3 is shown in Table 7. The length vector for this period is (l_3, l_2, l_1), however, no production is allowed since $\sigma_3(1) = \sigma_3(2) = \sigma_3(3) = 0$, therefore no setups occur in period 3. Then, the inventory is calculated: $s_{13} = 26-10 = 16$, $s_{21} = 7 - 7 = 0$, and $s_{31} = 7 - 7 = 0$.

Table 7: D_1 in practice for period 3.

Item	$X_3 - s_2$	W_3	RK_3
l_1	$(10) - 26$	1	1^{nd}
l_2	$(7) - 7$	2	2^{nd}
l_3	$(7) - 7$	1	3^{nd}

The above procedure decodes the individual (4.12) to the MPCSPs solution presented in Table 8. To obtain the fitness value of (4.12), we consider the objective function (3.1). Hence, the fitness value is the sum of 300 *(3 setup costs),* 33 *(used object costs) and* 3.48 *(item inventory costs), i.e.,* 338.48. △

Table 8: Solution description.

Items	$t = 1$			$t = 2$		$t = 3$
	s	A		s	A	s
l_1	29	2	0	26	0	16
l_2	12	1	0	7	0	0
l_3	0	0	1	7	1	0
x		17	1		13	

We next describe a variation of D_1 where the first Phase is obtained in a different way and new cutting pattern construction specifications are added to Phase two.

4.2.3.3 Decoder 2 (D_2)

For this decoder, the chromosome is encoded as a vector of item productions, X, items occurrences, W, items assignments, RK, and heuristic selection, HS. Just as with D_1, the first Phase will act only on vector X and have size $|P|(|T|-1)$. The second Phase maps on the W, RK, and HS vectors. The length of W, RK, and HS are the same and equal to $|P||T|$, where $|P|$ genes are used for each vector in each period, therefore, a total of $3|P||T|$ random keys are required for Phase 2. Hence, the length of an individual, χ, when considering D_2, is $|P|(|T|-1)+3|P||T|$.

Example 5.3.2. *In order to illustrate the decoder, we consider the same instance presented in Table 4 and the following random keys vector:*

$$\chi = \Big(.1,\quad .037,\quad .02,\quad .9,\quad .19,\quad .9$$
$$.3,\quad .28,\quad .05,\quad .07,\quad .4,\quad .51,\quad .2,\quad .63,\quad .3,$$
$$.14,\quad .28,\quad .05,\quad .7,\quad .4,\quad .51,\quad .62,\quad .33,\quad .3$$
$$.3,\quad .88,\quad .05,\quad .47,\quad .07,\quad .39,\quad .2,\quad .63,\quad .3,\Big) \tag{4.22}$$

In a similar fashion as with vector (4.12), the first line is used in the first Phase of the decoder, i.e. to obtain vector X, while the remaining lines are used in Phase 2 in the following way: the second line is used to obtain W and the third and fourth to obtain RK and HS, respectively. △

The gene position notation is similar to the one in D_1. We next describe how to decode the random keys vector.

- The Phase 1 is similar to the one in D_1, but now $X_t(i)$ may contain partial demands from the next periods in the following way:

$$X_t(i) = d_{it} + \left\lceil \chi(it) \sum_{\tau=t+1}^{|T|} d_{i\tau} \right\rceil . \tag{4.23}$$

Considering Example 5.3.2, the calculations to obtain X are given as follows:

1. the vector of future demands from each period is given by $(\underbrace{13,12,13}_{t=1}, \underbrace{10,7,7}_{t=2})$.

2. multiplying the above vector with the first line of χ derives the vector $(\underbrace{1.3,0.44,0.26}_{t=1}, \underbrace{9,1.33,6.3}_{t=2})$.

Hence, applying the ceil function to the above vector and adding the value of the demand from each item in each period, the vector X is given by: $(\overbrace{8,6,2}^{t=1}, \overbrace{12,7,13}^{t=2})$.

- The Phase 2 decodes the vectors W, RK and HS in the following way:

  1. The W vector is obtained in the same way as described in Equation (4.15) of D_1.

  2. The RK vector decode is done in the same way as described in D_1 by Equation (4.16).

  3. The last vector of the second Phase will select the heuristics to construct the cutting patterns. For each t, the random key $\chi(jt + 3PT)$ will determine the selected heuristic to construct the cutting pattern $j, j \in [1, 2, ...|P|]$, i.e. at most $|P|$ cutting patterns are generated via HS per period. The selection is according to the interval in which the random key is. We selected 6 disjoints intervals in $[0, 1]$, one for each heuristic. The interval length is based on the computational time of the heuristic (the heuristic that consumes more time will be associated to a smaller interval). With exception from heuristic 3, which is a knapsack problem solved by a commercial solver, the 5 heuristics are quite similar in terms of computational effort and therefore are all attributed to uniform disjoints intervals of size 0.9/5. The knapsack based heuristic is then attributed to last interval of size 0.1. For demonstration purpose, we consider the following intervals for each heuristic: $[0, .18)$ for H1, $[.18, .36)$ for H2, $[.36, .46)$ for H3, $[.46, .64)$ for H4, $[.64, .82)$ for H5 and $[.82, 1)$ for H6. For Example 5.3.2, the vector HS decoded from χ is given by: $(\overbrace{H2, H6, H1}^{t=1}, \overbrace{H4, H1, H3}^{t=2}, \overbrace{H1, H3, H1}^{t=3})$.

We note that vectors W and RK are part of two distinct heuristics, $H5$ and $H6$, respectively. Just as with Decoder 1, we define $\sigma_t : P \to \{0, 1\}$ such that $\sigma_t(i) = 1$ if $X_t(i) > 0$ while $\sigma_t(i) = 0$ otherwise.

- **Cutting pattern construction:** the construction Phase consists is selecting the heuristic in HS and construct the cutting patterns to satisfy the production from Phase 1.

The heuristics are described as follows:

(H1) **First fit decreasing**: consists in assign, in decreasing order of the items length, items of type i such that $X_t(i) > 0$, until the object length L is exceed or $a_i > X_t(i)$. The procedure is outlined in Algorithm 2.

Algorithm 2: First fit decreasing for period t.

Vector Input: X_t
Output: $a \in \mathbb{R}^{|P|}$

$a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$, sort lengths in decreasing order;
while ($\exists$ i *such that* $\sigma_t(i) = 1$ *and* $a(i) \leq X_t(i)$ *and* $L^{'} + l_i \leq L$)
  $L^{'} \leftarrow L^{'} + l_i$;
  $a(i) \leftarrow a(i) + 1$;
  Go to next item in the order;

(H2) **Demand sorting**: consists in assign, in decreasing order of X_t, items of type i such that $X_t(i) > 0$, until the object length L is exceed or $a_i > X_t(i)$. The procedure is outlined in Algorithm 3.

Algorithm 3: Demand sorting for period t.

Vector Input: X_t
Output: $a \in \mathbb{R}^{|P|}$

$a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$, sort X_t in decreasing order;
while ($\exists$ i *such that* $\sigma_t(i) = 1$ *and* $a(i) \leq X_t(i)$ *and* $L^{'} + l_i \leq L$)
  $L^{'} \leftarrow L^{'} + l_i$;
  $a(i) \leftarrow a(i) + 1$;
  Go to next item in the order;

(H3) **Knapsack optimization**: solve the following bounded knapsack problem:

$$\textbf{Max} \sum_{i \in P | X_t(i) > 0} a_i X_t(i)$$

subject to:

$$\begin{aligned} & a_1 l_1 + a_2 l_2 + ... + a_{|P|} l_{|P|} \leq L & \\ & a_i \leq X_t(i) & \forall i \\ & a_i \in \mathbb{Z}^+ & \forall i, \end{aligned}$$

(H4) **Adapted single setup algorithm**: based on the single setup algorithm of Mobasher and Ekici (2013), the heuristic is outlined in Algorithm 4.

Algorithm 4: Adapted single setup for period t.

Vector Input: X_t, $U \in \mathbb{R}^{|P|}$
Output: $a \in \mathbb{R}^{|P|}$
1 $a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$, $U(i) = 1 \; \forall \; i$;
2 Item= $argmax_{i|X_t(i)>0} \dfrac{X_t(i)}{U(i)}$;
3 **if** $L^{'} + l_{Item} \leq L$ *and* $a(Item) \leq X_t(Item)$
4 $\quad U(Item) \leftarrow U(item) + 1$;
5 $\quad L^{'} \leftarrow L^{'} + l_{Item}$;
6 $\quad a(Item) \leftarrow a(Item) + 1$;
7 $\quad$ Go to line 2;

(H5) **Random item occurrences**: items types i are assigned in decreasing order of their demand values in X_t with $W_t(i)$ occurrences in the cutting pattern. The heuristic is outlined in Algorithm 5.

Algorithm 5: Random item occurrences for period t.

Vector Input: X_t, W_t
Output: $a \in \mathbb{R}^{|P|}$
1 $a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$, sort X_t in decreasing order;
2 **while** *(*$\exists$ i *such that* $\sigma_t(i) = 1$ *and* $a(i) \leq X_t(i)$ *and* $L^{'} + W_t(i)l_i \leq L$ *)*
3 $\quad L^{'} \leftarrow L^{'} + W_t(i)l_i$;
4 $\quad a(i) \leftarrow a(i) + W_t(i)$;
5 $\quad$ Go to next item in the order;

(H6) **Random sorting**: items are assigned according to the new arrangement setting obtained from vector RK until the object capacity is exceeded or $a_i > X_t(i)$, for each item type i. The heuristic is outlined in Algorithm 6.

Algorithm 6: Random sorting for period t.

Vector Input: X_t, RK_t
Output: $a \in \mathbb{R}^{|P|}$
1 $a \leftarrow \mathbf{0}$, $L^{'} \leftarrow 0$;
2 **while** *(*$\exists$ i *such that* $\sigma_t(i) = 1$ *and* $a(i) \leq X_t(i)$ *and* $L^{'} + l_i \leq L$*)*
3 $\quad L^{'} \leftarrow L^{'} + l_i$;
4 $\quad a(i) \leftarrow a(i) + 1$;
5 $\quad$ Go to next item in the order;

- **Cutting pattern frequency:** after a cutting pattern j, $j \in [1, ..., |P|]$, is constructed, the frequency in which j is used in period t, x_{jt}, is determined by one of the ratios:

$$R_{ij} = \left\lceil \frac{X_t(i)}{a_{ijt}} \right\rceil, \tag{4.24}$$

for all i such that $a_{ijt} > 0$. To determine such ratio, we proceed as follows:

- Define constants MAX and F such that $MAX = \sum_{i,a_{ijt}>0} R_{ij}$ and $F = 0$.
- Starting from the smallest R_{ij}:
  1. $F = F + R_{ij}$
  2. if $\dfrac{F}{MAX} \geq 0.5$ take $x_{jt} = R_{ij}$.
     else add the next ratio.

We note that if 1 is considered rather than 0.5, the frequency calculation is done by taking the biggest of the ratios in Equation (4.24), which is the same as the cutting pattern frequency calculation of Decoder 1 (Equation (4.17)). The choose of the value 0.5 is seen as a mid term between overproduction by taking the biggest ratio and the minimum production by taking the smallest one.

We also highlight that if $j = |P|$ and $\exists\ i$ such that $X_t(i) > 0$, then the First Fit Decreasing heuristic (H1) is applied as many time as needed. When $X_t(i) \leq 0$ for all i, the construction process stops, the inventory variables are computed and X_{t+1} is updated accordingly. The framework for Decoder 2 considering period t is shown in Figure 13.

Next, an illustration of the construction Phase of D_2 considering the instance from Example 4.2.2 and individual (4.22) is presented.

Example 4.2.3. *(***Continuation***) Starting from period 1, the HS_1 decoded from Phase 2 states we construct the cutting patterns using heuristics $H2$, $H6$ and $H1$, in this order. The remaining decoded information regarding Phases 1 and 2 is shown in Table 9.*

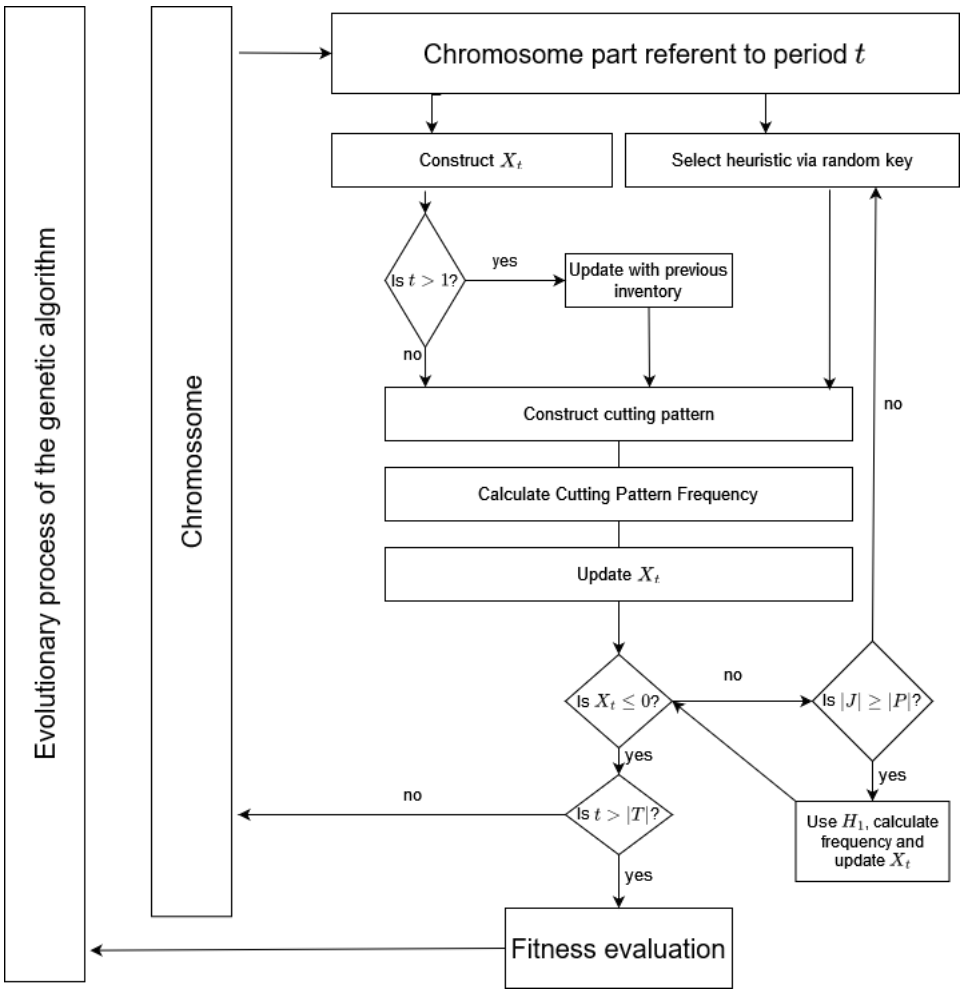
Table 9: D_2 in practice for period 1.

Item	X_1	W_1 (H5)	RK_1 (H6)
l_1	$6+2$	2	2^{nd}
l_2	$5+1$	1	1^{nd}
l_3	$1+1$	1	3^{nd}

Then, the period 1 *cutting pattern construction follows as:*

- *For $j = 1$, the chosen heuristic is $H2$, sorting X_1 in decreasing order gives the following item sequence: l_1, l_2 and l_3. By Algorithm 2, the cutting pattern $\vec{a}_1$ is given as $[2, 2, 0]$. To calculate the cutting pattern frequency, the ratios for items l_1 and l_2 are given, according to Equation (4.24), by 4 and 3, respectively. Then, since the fraction $3/MAX$ yields a*

Figure 13: D_2 framework for period t. Source: author.



value lesser than 0.5*, where* $MAX = 3 + 4$*, the frequency of the cutting pattern in period 1,* x_{11}*, is set as the largest and last ratio, i.e.,* 4*.*

The stock variables are updated as $s_{11} = 8 - 6 = 2$ *and* $s_{21} = 8 - 5 = 3$*. Similarly,* $X_1(1)$ *and* $X_1(2)$ *are updated by* $8 - 8 = 0$ *and* $6 - 8 = -2$*, respectively.*

- *For* $j = 2$*, the H6 heuristic is the chosen one. Note that* $X_1(3)$ *is the only positive value of* X_1*. Therefore, By Algorithm 6, the cutting pattern is* $\vec{a}_2 = [0, 0, 1]$*. Since only one ratio is obtained from* $\vec{a}_2$*, the cutting pattern frequency in period 1,* x_{21}*, will be given directly by Equation (4.24), i.e.,* $x_{21} = X_1(3)/1 = 2$*. The inventory variable* s_{31} *is updated by* $2 - 1 = 1$*.*

- *For* $j = 3$*, since no more production is allowed, the construction for period* 1 *stops.*

Next, HS_2 *gives the following heuristics to be applied: H4, H1 and H3. The remaining decoded information for period* 2 *is given in Table 10. Let us proceed to the cutting patterns construction.*

Table 10: D_2 in practice for period 2.

Item	$X_2 - s_1$	W_2 (H5)	RK_2 (H6)
l_1	$(3 + 9) - 2$	1	1^{nd}
l_2	$(5 + 2) - 3$	1	3^{nd}
l_3	$(6 + 7) - 1$	1	2^{nd}

- *For* $j = 1$*, we use H4 to obtain the cutting pattern, as Algorithm 4 suggests, we need the vectors* $X_2 = (10, 4, 12)$ *and* $U = (1, 1, 1)$*. The first item type* i *to be assigned is the one with biggest ratio* $X_2(i)/U(i)$*, which is* l_3*, then* $U(3)$ *receives the value* 2 *and we restart the process. The next item to be assigned would be item* 1 *with ratio equal to* 10*, then* $U(1) = 2$*. Since* $l_3 + l_1 = 9$*, no more items are allowed into the cutting pattern and therefore, the construction process stops. The cutting pattern* $\vec{a}_3$ *is given by* $(1, 0, 1)$*. According to Equation (4.24), the ratios for items* 1 *and* 3 *are* 10 *and* 12*, respectively. Since* $10/MAX < 0.5$*, where* $MAX = 10 + 12$*, the frequency,* x_{32}*, is given as* 12*.*

  Now, the inventory variables s_{12} *and* s_{32} *are updated by* $12 - 3 + s_{11} = 11$ *and* $12 - 6 + s_{31} = 7$*, respectively.*

- *For* $j = 2$*, the H1 heuristic is selected. Since only* $X_2(2) > 0$*, The heuristic will simply repeat* l_2 *until the object capacity is exceeded. Hence,* $\vec{a}_{42} = (0, 3, 0)$*. Following Equation (4.24), the frequency* x_{42} *is set to be* $\lceil 4/3 \rceil = 2$*. The inventory variable,* s_{22}*, is set as* $6 + s_{21} - 5 = 8$*.*

- *For $j = 3$, no production is needed, therefore the construction process is halted.*

The decoded information for period 3 *is given in Table 11. As we can note, no cutting patterns are required in this period since* $X_3 - s_2 \leq 0$.

Table 11: D_2 in practice for period 3.

Item	$X_3 - s_2$	W_3 (H5)	RK_3 (H6)
l_1	$(10) - 11$	2	1^{nd}
l_2	$(7) - 8$	1	2^{nd}
l_3	$(7) - 7$	1	3^{nd}

The fitness value of χ *according to* D_2 *is* 421.24 *(*400 *(from setup costs)+* 20 *(from used objects costs)+* 1.24 *(from items inventory costs)). The obtained solution is detailed in Table 12.* △

Table 12: Solution description.

Items	$t = 1$			$t = 2$			$t = 3$
	s	A		s	A		s
l_1	2	2	0	11	1	0	1
l_2	3	2	0	7	0	3	1
l_3	1	0	1	8	1	0	0
x		4	2		12	2	

CHAPTER 5

COMPUTATIONAL STUDY

The main objectives of this chapter are to complement the theoretical results presented earlier by providing computational insights to address questions, such as "how difficult is a formulation to solve?" and "how much better is a formulation?" in a practical context with a range of classes of test problems, and to evaluate the robustness of the proposed *ABRKGA* when compared to the column generation technique. The tests were tested with randomly generated instances in a PC with Intel Core i7 3.2 GHz CPU processor and 16 GB of RAM. The mathematical formulations were implemented in Visual Studio 2017 using the C callable library of IBM ILOG Cplex 12.9.0. In order to evaluate the impact of the proposed formulations and facility location reformulations of the *MPCSPs* we analyzed two sets of instances, as presented in the next sections. In the first set, some easy instances are considered by varying item and period number, object size and inventory cost, while in the second set, more challenging instances are considered by considering smaller values for the item length.

In Section 5.1, the pattern-based and the pseudo-polynomial formulations are exactly solved by a commercial optimization package with a default *MIP* tolerance and a time limit of 1800 seconds. In order to obtain an exact solution to this set of easy instances, the pattern-based formulations (*AGG* and *AGGFL*) consider all the possible columns, i.e., all possible cutting patterns are generated a priori (we note the time spent in the generation process is small and is not accounted for in the computational results) and considered when solving the linear relaxation and the mixed-integer cutting stock problem. We also note that, for this set of solution, *AREFL** is used in computational testing (rather than *AREFL*). In Section 5.2, more challenging instances are considered and the

pattern based formulations *AGG* and *AGGFL* are solved by a column generation-based heuristic procedure called, respectively, *AGG-H* and *AGGFL-H*. In Section 5.3 a sensitivity analysis of the lower bounds regarding the object cost is conducted considering the second set of instances. Finally, in Section 5.4 the computational experiments considering the *ABRKGA* using the second set of instances is presented. The *ABRKGA* is compared using the column generation technique. The chapter is then concluded with computational results considering a combined method using the *ABRKGA* and the column generation procedure.

We evaluate and compare the performance of the proposed models with respect to different factors, such as *LP* relaxation solution, Z_{LP}, time to solve the *LP* relaxation, T_{LP} (in seconds), mixed-integer problem (*MIP*) solution, Z_{IP}, time to solve the *MIP*, T_{IP} (in seconds), absolute *MIP* gap, GAP_A (in percentage), and relative *MIP* gap, GAP_R (in percentage). The absolute and relative gaps are calculated according to equations (5.1) and (5.2), respectively, where Z_{BestLB} is the best lower bound achieved when solving the integer problem. We consider the value GAP_R as the one provided by the optimization package, after solving the integer problem.

$$GAP_A = \frac{100(Z_{IP} - Z_{LP})}{Z_{IP}} \tag{5.1}$$

$$GAP_R = \frac{100(Z_{IP} - Z_{BestLB})}{Z_{IP}} \tag{5.2}$$

In order to find solutions with the lowest number of different cutting patterns, the value per stock object, *oc*, is considered as 1 when solving the *MIP* problems in Sections 5.1 and 5.2.

5.1 Computational results for the exact solution approaches

The first data set used to generate instances for the multi-period cutting stock problem with setups is based on data from the literature, more specifically, Poldi and de Araujo (2016) (number of periods, number of items and item inventory cost), Cui et al. (2015) (setup cost) and the CUTGEN1 generator proposed by Gau and Wäscher (1995) (length of items and demand of items per period), and with specifics as follows:

- number of periods: $|T| = \{3, 6\}$;
- number of items: $|P| = \{10, 20\}$;

- object length: $L = \begin{cases} 500, & \text{Small;} \\ 1000, & \text{Medium;} \\ 1500, & \text{High.} \end{cases}$
- setup costs: $sc_t = 100$ for all t;
- item length: $l_i \in [0.375L, 0.625L]$;
- inventory costs: $h_{it} = \begin{cases} 0.01l_i, & \text{Low;} \\ 0.05l_i, & \text{High.} \end{cases}$
- item demand: Generated by CUTGEN1[1], with $d = 10$.

Using a full factorial design, the data set consists of 2 x 2 x 3 x 2= 24 problem classes. For each class, 20 instances are generated, resulting in a total of 480 problems. It is worth highlighting that these instances are easy to solve: requirements consist of less than 20 items of length greater than $L/3$, which implies less than 2 items per object. So basically, we do not need more than 400 cutting patterns for complete enumeration. Additionally, considering object lengths 500, 1000 or 1500 does not change the maximum number of possible subset sums.

Table 13 presents the average number of variables for classes of instances with medium object length (160 instances). The *AGG* model presents the smaller number of variables for these classes, followed by its reformulation, *AGGFL*, with a 133% overall increment when compared to *AGG*. When the number of items is increased from 10 to 20 the *ARE* formulation presents an increase of variables of 137%, followed by 291% for the *AJS* and *AVR* models and 300% of increase for the *AGG* model. We highlight that the *AREFL** formulation presents the lowest overall variable increment (only 2.65%) when compared to its original formulation *ARE*, followed by an overall increment of 15.93% for the *AVRFL* model when compared to *AVR* model.

Table 13: Average number of variables order for classes of instances with $L = 1000$.

Classes	Instances average size							
($\|T\|/\|P\|$)	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*
Classes (3/10)	900	1747.2	2907.3	3519.4	1500	3494.4	2997.8	4119.9
Classes (3/20)	3600	6559.5	6896	13,144	6000	13,119	7096	15,544
Classes (6/10)	1800	6650.4	10,606	13,325.8	4800	23,276.4	11,006	15,425.8
Classes (6/20)	7200	26,353	21,693.5	52,731.4	19,200	92,236.2	22,397.5	61,131
Mean	**3375**	**10,327.52**	**10,593.17**	**20,680.15**	**7875**	**33,031.5**	**10,874.32**	**24,055.17**

[1]CUTGEN1 is an instance generator for the one-dimensional cutting stock problem (Gau and Wäscher (1995))

Table 14 displays, for each class and mathematical model, the average values for the *LP* relaxation and computational time to solve the *LP*. The two best lower bounds for each class are highlighted in red. For all classes, the facility location reformulations present considerable improvements of the lower bound when compared to the original formulation, which can reach up to 339% (Class 24 with the *AJSFL*). The *AJSFL* presents the highest improvements of the lower bound when compared to its original formulation (*AJS*), reaching on average 235.78% increase of the lower bound, followed by the *AGGFL* with 189% average improvement. The *AVRFL* reformulation presents an average improvement of 164% and the *AREFL** reformulation presents an average improvement of 16.58%. Containing the largest problems in this data set, Group 6/20 of instances (Classes 19 to 24) demonstrate the best lower bounds obtained by the reformulations when compared with their respective original formulations. It is worth to mention that it is known in the literature that *AGG* and *ARE* formulations provide the same lower bound quality, and we have shown in this study (Theorem 3.1) that the same behavior holds for the facility location reformulations *AGGFL* and *AREFL*. However, since we are using the alternative *AREFL** described in Section 3.5.3.1, the lower bound values are not equal, whereas the *AREFL** attain smaller lower bounds than *AGGFL* and greater values than *AGG* for the specific cost configuration. For all classes, *AJS* consistently obtained the worst lower bounds, whereas *AGGFL* consistently provided the best values. The facility location reformulation bounds are highly affected by the inventory holding cost when compared to its original formulation. The computational time required to solve the *LP* relaxation is relatively small (not more than 0.75 seconds on average), and the original formulations are naturally faster to solve than their respective facility location reformulations.

Table 15 displays, for each mathematical model and class, the objective function value found for the *MIP* and the time spend to solve the *MIP*. The two best integer solutions for each class are highlighted in red. Note that *AJS* and *AJSFL* are not able to solve some instances in Group 6/20 of instances (Classes 19, 20, 21, 23 and 24), as remarked in the table, therefore, we are not able to compare these two formulations in these classes. For the *AVRFL* the results are even worse, since it could not solve all instances from Group 6/20, however its original formulation (*AVR*) obtained feasible solution for all classes. The results show that, for Group 3/10 of instances (Classes 1 to 6), *AGG*, *ARE*, *AGGFL*, and *AREFL** present similar objective function values. However, as the problem size increases due to the number of period and items (Classes 7 to 24), the reformulations, on average, are able to find the best solutions in 13 classes, whereas the formulations found only in 5 classes. This fact reinforces the motivation for using reformulations when solving the

Table 14: Evaluation of the lower bounds obtained by the proposed mathematical models

Classes	LP relaxation solution (Z_{LP})								Time for the LP relaxation (seconds) (T_{LP})							
($\|T\|/\|P\|/L/h_{it}$)	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S/L)	690.18	497.48	690.18	648.89	1334.83	1080.84	795.92	1216.1	0.002	0.008	0.02	0.07	0.061	0.022	0.025	0.28
Class 2 (3/10/S/H)	691.49	497.47	691.49	650.04	1757.47	1340.96	839.82	1526.66	0.002	0.01	0.01	0.13	0.04	0.08	0.025	0.28
Class 3 (3/10/M/L)	825.22	551.14	825.22	768.47	1756.82	1325.57	942.06	1568.86	0.003	0.007	0.015	0.08	0.04	0.08	0.026	0.3
Class 4 (3/10/M/H)	825.22	551.14	825.22	768.91	2082.26	1502.31	975	1809.12	0.002	0.018	0.01	0.14	0.02	0.1	0.2	0.28
Class 5 (3/10/H/L)	661.61	474.23	661.61	610.39	1619.65	1268.46	795.94	1409.22	0.002	0.012	0.02	0.13	0.036	0.063	0.03	0.38
Class 6 (3/10/H/H)	661.61	474.22	661.61	640.39	1849.15	1408.52	821.71	1569.54	0.002	0.02	0.02	0.19	0.036	0.08	0.04	0.4
Mean (3/10)	**725.89**	**507.61**	**725.89**	**681.18**	**1733.36**	**1321.11**	**861.74**	**1516.53**	**0.002**	**0.0125**	**0.016**	**0.13**	**0.039**	**0.07**	**0.058**	**0.32**
Class 7 (3/20/S/L)	1177.77	862.74	1177.77	1093.4	2485.17	2089.64	1434.16	2331.85	0.008	0.053	0.037	0.7	0.45	0.17	0.08	2.67
Class 8 (3/20/S/H)	1179.16	862.76	1179.16	1093.48	3264.98	2632.75	1507.3	2872.73	0.005	0.08	0.03	1.25	0.26	0.32	0.067	2.49
Class 9 (3/20/M/L)	1386.66	923.69	1386.66	1114.7	3333.74	2596.16	1750	2978.92	0.07	0.075	0.036	0.7	0.173	0.256	0.07	3.1
Class 10 (3/20/M/H)	1386.66	923.70	1386.66	1144.55	3982.45	2976.33	1816.97	3412.15	0.003	0.09	0.03	1.56	0.15	0.41	0.06	2.65
Class 11 (3/20/H/L)	1241.33	895.12	1241.33	1159.78	3152.64	2550.42	1538.87	2798.73	0.009	0.074	0.05	1.14	0.3	0.263	0.9	3.61
Class 12 (3/20/H/H)	1241.33	895.12	1241.33	1168.15	3558.46	2812.13	1576	3066.46	0.003	0.1	0.05	1.86	0.022	0.4	0.1	3.96
Mean (3/20)	**1268.82**	**893.85**	**1268.82**	**1195.68**	**3296.24**	**2609.57**	**1341.22**	**2910.14**	**0.016**	**0.079**	**0.039**	**1.2**	**0.22**	**0.3**	**0.24**	**3.1**
Class 13 (6/10/S/L)	1140.65	780.11	1140.65	1063.67	2590.17	2096.32	1278	2353.44	0.003	0.048	0.05	0.31	0.18	0.21	0.143	1.37
Class 14 (6/10/S/H)	1144.34	780.1	1144.34	1066.77	3556.75	2705.33	1325.71	3078.552	0.005	0.11	0.04	0.65	0.09	0.43	0.09	1.21
Class 15 (6/10/M/L)	1263.59	810.28	1263.59	1193.96	3411.78	2601.15	1461.2	3064.51	0.04	0.06	0.057	0.37	0.085	0.27	0.114	1.48
Class 16 (6/10/M/H)	1263.59	810.28	1263.59	1194.02	4215.42	3038.16	1489.21	3636.96	0.003	0.1	0.05	0.67	0.07	0.55	0.12	1.34
Class 17 (6/10/H/L)	1084.15	748	1084.15	1009.11	3183.22	2526.84	1267.32	2817.29	0.004	0.111	0.08	0.6	0.13	0.335	0.12	1.76
Class 18 (6/10/H/H)	1084.15	748	1084.15	1009.11	3706.36	2819.8	1291.54	3161.36	0.004	0.14	0.07	1.01	0.09	0.59	0.13	1.49
Mean (6/10)	**1163.42**	**779.46**	**1163.42**	**1089.44**	**3443.95**	**2631.27**	**1352.16**	**3018.69**	**0.009**	**0.095**	**0.056**	**0.6**	**0.1**	**0.4**	**0.12**	**1.49**
Class 19 (6/20/S/L)	1783.48	1165.56	1783.48	1688.4	4683.41	3914.86	2213.24	4332.42	0.011	0.19	0.18	4.3	1.34	1.16	1.5	12
Class 20 (6/20/S/H)	1784.1	1190.5	1784.1	1654.2	6375.16	5189.91	2292.42	5730.15	0.014	0.43	0.18	11	0.9	2.48	0.85	12.79
Class 21 (6/20/M/L)	2334.79	1480.64	2334.79	2166.55	6538.44	5071.75	2740.41	5953.73	0.072	0.29	0.098	4.2	0.66	1.455	0.3	13.58
Class 22 (6/20/M/H)	2334.79	1437.58	2334.79	2166.55	8006.94	5941.31	2801.64	7082.36	0.007	0.47	0.09	8.1	0.43	3.16	0.23	15.26
Class 23 (6/20/H/L)	1901.43	1226	1901.43	1780.75	6093.52	4958.93	2358.74	5465.75	0.009	0.343	0.14	5.76	0.945	1.882	0.58	16.9
Class 24 (6/20/H/H)	1901.43	1272.47	1901.43	1780.75	7028.42	5589.78	2391.7	6148.34	0.011	0.7	0.17	7.99	0.61	3.21	0.57	17.16
Mean (6/20)	**2006.67**	**1295.46**	**2006.67**	**1872.87**	**6434.53**	**5111.09**	**2466.35**	**5785.46**	**0.02**	**0.4**	**0.143**	**6.9**	**0.81**	**2.22**	**0.51**	**14.62**
Mean	**1291.2**	**869.1**	**1291.2**	**1199.6**	**3731.97**	**2918.26**	**1505.36**	**3320.96**	**0.01**	**0.15**	**0.06**	**2.22**	**0.3**	**0.75**	**0.22**	**4.87**

problem. The *AREFL** reformulation presents the highest improvements when compared to its original formulation *ARE*. The *AJS* presents the worse values to the problem, followed by *AVR*, *AVRFL* and *AJSFL*. Among the reformulations the *AVRFL* and *AJSFL* were the only one whose computation performance was slightly worse then its original formulations. There is no clear behavior of the formulations regarding the computational time, only the natural fact that as the problem enlarges, more computational time is spent to solve the mixed-integer problem. For most of the classes, the reformulations are faster than their respective original formulations. We note that the average solution time for the knapsack-based formulations almost reaches its limit even for the smallest test instances with three periods and ten items. The smallest average computational time is presented by *AGGFL* (932.94 seconds), followed by *AGG* (981.94 seconds) and *AREFL** (1004.79 seconds).

In Table 16, we present the average values for the absolute and relative gaps obtained by each formulation in each class within the time limit of 1800 seconds. The best absolute and relative gaps for each class are highlighted in red and blue, respectively. As noted earlier, *AJS* and *AJSFL* could not solve some instances in Classes 19, 20, 21, 23 and 24, while *AVRFL* could not solve some instances from all Group 6/20 and hence, we are not able to compare these three formulations in these classes. The absolute gap obtained by the reformulations are considerably smaller when compared to the original formulations, with *AGGFL* achieving the smallest average values (10.23%). The impact of the problem size on the absolute gap can be seen more dramatically for formulations, while gaps for reformulations may be even observed to decrease. Considering relative gaps (i.e., the final gap value provided by the optimization package), *AGGFL* and *AREFL** reformulations are able to find the smallest gaps on all classes except for Classes 14 and 16, where *AGG* found better results. These results are encouraging to justify the use of extended reformulations in cutting stock problems, both in terms of obtaining much stronger lower bounds and also directing the search effectively to high quality solutions of the problem.

Table 15: Evaluation of the upper bounds (feasible solutions) obtained by the proposed mathematical models.

Classes	*MIP* solution (Z_{IP})								Time for the *MIP* solution (seconds) (T_{IP})							
($\|T\|/\|P\|/L/h_{it}$)	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S/L)	1437.24	1438.1	1441.53	1448.29	1439.49	1441.32	1439.49	1447.1	10.93	1313.81	359.44	1513.31	6.07	1551	161.62	1798
Class 2 (3/10/S/H)	1985.66	2007.77	1989.5	2124.89	1985.66	2008.05	1985.66	2003.81	26.1	1661.7	586.75	1721	11.97	1793	224.8	1712.51
Class 3 (3/10/M/L)	1874.08	1875.93	1877.12	1885.46	1874.86	1877.02	1874.86	1876.97	4.11	1473.51	953.51	1608.77	2.5	1706	82.35	1501
Class 4 (3/10/M/H)	2359.36	2375.12	2359.36	2464.47	2359.36	2368	2359.36	2384.24	5	1800	35.77	1800	7	1712	18.4	1800
Class 5 (3/10/H/L)	1795.53	1807.34	1796.1	1866.4	1796.21	1802.8	1796.21	1803.92	26.55	1625.74	668.23	1692.4	15.1	1789	186.89	1774.56
Class 6 (3/10/H/H)	2233.85	2301.18	2233.9	2515.17	2233.9	2286.53	2233.9	2290.35	269.66	1800	301.64	1800	282.43	1792	118.16	1800
Mean (3/10)	**1947.62**	**1967.57**	**2155.2**	**2050.78**	**1948.55**	**1963.95**	**1948.25**	**1967.73**	**56.95**	**1612.46**	**475.79**	**1690.35**	**54.18**	**1723.33**	**132.03**	**1730.84**
Class 7 (3/20/S/L)	2645.96	2805.84	2677.53	2987.56	2642.06	2712.83	2648.13	2853.56	1482.88	1800	1322.8	1800	1001.87	1800	1023.841	1800
Class 8 (3/20/S/H)	3628.53	4275.3	3668.1	4590.22	3626.94	3789.4	3619.4	4138.43	1666.36	1800	1481.31	1800	1618.07	1800	1308.8	1800
Class 9 (3/20/M/L)	3519.58	3761.69	3661.2	3869.48	3516.96	3597.01	3518.04	3737.37	798.73	1800	1570.74	1800	417.89	1800	470.93	1800
Class 10 (3/20/M/H)	4419.78	5115.2	4403.85	5458.26	4417.93	4582.19	4402.81	4906.97	1188.14	1793	547.27	1800	1171.12	1796	524.83	1800
Class 11 (3/20/H/L)	3400.57	3854.08	3452.39	4197.44	3397.06	3488.2	3400.97	3758	1387.615	1800	1391.23	1800	1303.2	1800	782.69	1800
Class 12 (3/20/H/H)	4181.97	5124.53	4120.76	5838.13	4168.01	4415.48	4115.91	5229.51	1762.23	1800	985.77	1800	1795	1800	882.65	1800
Mean (3/20)	**3692.73**	**4156.11**	**3644.7**	**4490.18**	**3628.16**	**3764.18**	**3617.54**	**4103.97**	**1380.2**	**1800**	**1216.52**	**1800**	**1217.86**	**1800**	**832.29**	**1800**
Class 13 (6/10/S/L)	2751.34	2891.51	2850.63	3008.19	2759.1	2784.95	2787.51	2831.01	667.8	1800	1736.44	1800	379.12	1800	1453.279	1800
Class 14 (6/10/S/H)	3966.1	4346.57	4063.34	5024.04	3968.88	4064.04	3996.17	4129.94	996.38	1800	1756.18	1800	1056.38	1800	1545.53	1800
Class 15 (6/10/M/L)	3638.02	3844.86	3728.78	3978.76	3639.83	3672.543	3658.28	3713.54	289.05	1800	1707.31	1800	238.24	1715	1347.28	1746
Class 16 (6/10/M/H)	4761.7	5169.94	4775.92	5605.14	4761.68	4902.62	4772.42	4921.16	416.44	1800	1225.53	1800	610.87	1714	987.1	1712
Class 17 (6/10/H/L)	3520.79	3804.11	3694.85	4386.26	3519.35	3617.17	3570.02	3724	966.97	1800	1685.3	1800	910.42	1800	1427.35	1800
Class 18 (6/10/H/H)	4497.37	5139.88	4543.85	6258.22	4510.18	4724.46	4510.54	4924.84	1398.83	1800	1357.4	1800	1448.75	1800	1247.2	18000
Mean (6/10)	**3855.89**	**4199.48**	**3942.9**	**4710.1**	**3859.84**	**3960.96**	**3882.49**	**4040.76**	**789.24**	**1800**	**1578.03**	**1800**	**773.96**	**1771.5**	**1334.62**	**1777.36**
Class 19 (6/20/S/L)	5114.07	36046.3 [1]	6058.94	6824.32	5062.43	5462.07	5217.38	6729.13 [4]	1800	1800	1800	1800	1800	1800	1800	1800
Class 20 (6/20/S/H)	7266.47	7.27×10^6 [2]	8490.57	10757	7220.55	8260.91	7453.32	9813.57 [4]	1800	1800	1800	1800	1800	1800	1767.26	1800
Class 21 (6/20/M/L)	6916.9	8258.1 [1]	7642.46	8554.7	6886.58	7270	6978.96	8352.82 [3]	1711.3	1800	1793.58	1800	1711.22	1800	1722.58	1800
Class 22 (6/20/M/H)	9006.55	3×10^5	9018.74	11768.58	9017.12	10066.8	8913.74	13369.83 [3]	1708.3	1800	1766.17	1800	1719.2	1800	1707.34	1800
Class 23 (6/20/H/L)	6777.98	9731.54 [3]	7862.44	8485.73	6653.63	7303.25	6840.87	9542.9 [1]	1800	1800	1800	1800	1800	1800	1800	1800
Class 24 (6/20/H/H)	8545.51	4×10^7 [2]	8767.3	8796.71	8703.31	10630.6 [3]	8312.26	15899.15 [1]	1800	1800	1800	1800	1800	1800	1766.42	1800
Mean (6/20)	**7271.25**	**7.9×10^6**	**7973.41**	**9197.85**	**7257.27**	**8165.60**	**7286.09**	**10617.9**	**1769.93**	**1800**	**1793.29**	**1800**	**1771.74**	**1800**	**1760.6**	**1800**
Mean	**4162.94**	**2.1×10^6**	**4424**	**5112.23**	**4158.73**	**4452.34**	**4169.38**	**5301.66**	**981.94**	**1750.77**	**1260.39**	**1772**	**932.94**	**1773**	**1004.79**	**1783.23**

[1] 15 feasible solutions were found; [2] 9 feasible solutions were found; [3] 17 feasible solutions were found. [4] 16 feasible solutions were found.

Table 16: Evaluation of the absolute and relative Gaps obtained by the proposed mathematical models.

Classes	Absolute Gap (Gap_A) (%)								Relative Gap (Gap_R) (%)							
($\|T\|/\|P\|/L/h_{it}$)	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*	*AGG*	*AJS*	*ARE*	*AVR*	*AGGFL*	*AJSFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S/L)	52.28	65.33	52.32	55.58	7.51	24.8	44.75	16.31	0.006	2.86	0.43	5.16	0.005	6.9	0.05	5.86
Class 2 (3/10/S/H)	65.9	75.22	65.46	69.57	12.31	33.1	57.74	24.26	0.008	10.7	0.67	18.43	0.007	16.1	0.006	10.6
Class 3 (3/10/M/L)	56.18	70.41	56.18	59.55	6.48	28.91	49.9	16.78	0.006	4.32	0.17	4.78	0.005	8.37	0.004	4.4
Class 4 (3/10/M/H)	65.23	76.89	65.24	69	12	36.29	59	24.6	0.005	12.01	0.005	15.36	0.005	14.86	0.007	10.17
Class 5 (3/10/H/L)	63.49	73.8	63.5	67.67	10.1	29.41	55.51	22.36	0.007	7.66	0.51	12.82	0.006	12.22	0.03	8.54
Class 6 (3/10/H/H)	70.6	79.45	70.6	75.72	17.54	38.3	63	31.82	0.14	19.6	0.2	28.12	0.23	21.31	0.23	18.77
Mean (3/10)	**62.28**	**73.5**	**62.22**	**66.18**	**10.99**	**31.8**	**54.98**	**22.69**	**0.029**	**9.52**	**0.31**	**14.11**	**0.043**	**13.29**	**0.017**	**9.75**
Class 7 (3/20/S/L)	55.57	69.22	56.05	62.72	6	22.9	45.68	18.37	2	18.17	2.6	19.45	0.85	15.04	1.05	15.22
Class 8 (3/20/S/H)	67.5	79.79	67.86	74.91	9.97	30.44	58.22	30.43	3.4	31.42	3.5	34.1	2.63	19.34	2.51	26.43
Class 9 (3/20/M/L)	60.81	75.42	60.88	66.69	5.26	27.73	50.33	20.3	0.57	20.46	3.98	17.55	0.037	13.92	0.26	13.75
Class 10 (3/20/M/H)	68.74	81.73	68.62	75.92	9.94	35	60.76	30.38	1.67	29.35	0.27	31.66	1.6	18.65	0.38	22.66
Class 11 (3/20/H/L)	63.6	76.78	64.13	79.66	7.29	26.86	54.68	25.48	2.18	26.92	3.23	29.4	1.15	15.8	1.34	21.16
Class 12 (3/20/H/H)	70.39	82.48	69.87	79.66	14.7	36.25	61.55	40.91	5.5	37.41	3.94	44	5.3	25.1	3.94	37.52
Mean (3/20)	**64.43**	**77.57**	**64.57**	**72**	**8.86**	**29.86**	**64.33**	**29.4**	**2.55**	**27.29**	**1.92**	**27.64**	**1.93**	**17.98**	**1.58**	**22.8**
Class 13 (6/10/S/L)	58.65	72.97	60.06	64.71	6.23	24.6	54.13	17.1	0.39	16.82	5.94	16.75	0.072	9.46	1.66	8.85
Class 14 (6/10/S/H)	71.22	82.02	71.9	78.81	10.58	33.34	68.2	25.75	0.9	26.4	4.87	43.71	0.97	17.83	2.53	14.31
Class 15 (6/10/M/L)	65.42	78.88	60.07	70.11	6.38	28.86	60.17	17.82	0.089	19.22	3.5	17.98	0.056	8.7	1.53	8.18
Class 16 (6/10/M/H)	73.56	84.23	73.64	78.75	11.65	37.91	68.88	26.47	0.25	26.38	1.16	38.47	0.38	18.73	0.86	14.74
Class 17 (6/10/H/L)	69.4	80.31	70.8	77	9.73	30	64.4	24.65	0.64	22.82	7.25	37.02	0.16	13.91	1.86	14.9
Class 18 (6/10/H/H)	76.01	85.43	76.14	83.32	18.1	40.31	71.27	40.91	2.66	34.73	3.27	58.64	3.94	26.4	2.12	37.52
Mean (6/10)	**69.04**	**80.64**	**68.77**	**75.46**	**10.44**	**32.5**	**64.51**	**24.6**	**0.82**	**24.39**	**4.37**	**35.43**	**0.92**	**15.84**	**1.76**	**14.3**
Class 19 (6/20/S/L)	65.46	81.2 [1]	70.22	75.23	7.58	28.25	57.44	[4] 34.23	6.56	33.8	20.1	40.63	4.1	20.45	6.93	33.62
Class 20 (6/20/S/H)	75.53	92.34 [2]	78.8	84.16	11.85	37.13	69	[4] 40.22	7.24	62.64	19.13	46.1	6.4	30.64	7.59	39.29
Class 21 (6/20/M/L)	66.32	81.98 [1]	68.82	74.37	5.16	30.1	60.86	[3] 28.48	3.1	27.63	9.98	29.11	2.33	18.46	2.94	25.81
Class 22 (6/20/M/H)	74.13	88.19	74.16	81	11.38	40.92	68.63	[3] 41.12	5.1	42.19	3.38	36.24	5.63	29.86	2.1	38.62
Class 23 (6/20/H/L)	72.1	85.59 [3]	75.72	78.21	8.56	32.1	65.47	[1] 41.53	6.47	43.17	18.6	35.29	4.39	24.8	6.92	40.64
Class 24 (6/20/H/H)	77.79	95.22 [2]	78.31	79.42	19.31	47.25 [3]	72	[1] 55.02	9.97	77.47	8.48	37.5	2.35	41.84	3.86	54.17
Mean (6/20)	**71.89**	**87.42**	**74.33**	**78.84**	**10.64**	**35.96**	**65.57**	**40.1**	**6.41**	**47.82**	**13.28**	**37.47**	**4.2**	**27.68**	**5.05**	**38.7**
Mean	**66.91**	**79.79**	**67.47**	**73.13**	**10.23**	**32.53**	**62.45**	**28.97**	**2.45**	**27.26**	**4.97**	**29.09**	**1.77**	**18.7**	**2.11**	**21.83**

[1] 15 feasible solutions were found; [2] 9 feasible solutions were found; [3] 17 feasible solutions were found. [4] 16 feasible solutions were found.

5.2 Computational results for the exact and heuristic solution approaches

In order to better evaluate the *MIP* performance of the formulations proposed in this thesis, the mathematical models were tested on difficult instances. To this aim, in the second data set used to generate instances, we only vary the item length (generated based on Gau and Wäscher (1995)) while keeping the object size ($L = 1000$), the inventory costs (low) and the setup cost as constant. Hence, the parameter specifics are as follows:

- number of periods: $|T| = \{3, 6\}$;
- number of items: $|P| = \{10, 20\}$;
- item length: $l_i \in \begin{cases} [0.01, 0.2]L, & \text{Small;} \\ [0.01, 0.8]L & \text{Medium;} \\ [0.2, 0.8]L, & \text{High.} \end{cases}$

Different from Section 5.1, the time spent during the column generation procedure is accounted for in the total time spent by the models, T_{IP}.

Tables 17 and 18 display the average relative gap and number of feasible solutions obtained from each formulation, respectively. Since *AGG-H* and *AGGFL-H* are heuristic methods, their relative gaps are obtained via formula (5.2), where Z_{BestLB} is now the best lower bound of the *ARE* formulation for each respective instance. If the *ARE* model do not find feasible solution, then the relative gaps of *AGG-H* and *AGGFL-H* are considered as the absolute gap (Equation (5.1)) for all instance class. For instances from Group 3/10, the *AJS* formulation obtained the best gap, followed by *ARE*, *AREFL** and *AREFL*, respectively. Except for *AVRFL* in Class 1, the remaining models could find feasible solutions for all classes of Group 3/10. For instances from Group 3/20, only *AGG-H*, *AGGFL-H* and *ARE* could find all feasible solutions for Class 4, which contains the smallest length of items. The gap from the models solved heuristically in Class 4 were significantly smaller than the gap from *ARE* indicating a struggle of the arc-flow models when the number of items duplicate. We also note that the *AREFL** obtained 3 more feasible solutions than *AREFL* for the Class 4. As for Classes 5 and 6, the *AREFL** obtained the best average for relative gaps. Next, starting the analysis in Group 6/10, we note that in Class 7 all original models could find feasible solutions while, except from $AGGFL - H$ and *AJSFL* (which found only 2 feasible solutions), the facility location reformulations did not find feasible solutions. On overall, *AJS* obtained the best average

gap for all Group 6/10, while ARE obtained the best gaps only in Classes 8 and 9. In Group 6/20, only the heuristically solved models could find all feasible solutions for Class 10. The same was true for Class 11, even though ARE and AVR cold find 19 feasible solutions. As for Class 12, only $AVRFL$ could not find all feasible solutions. In general, besides the *AGG-H* and *AGGFL-H*, the models which obtained more feasible solution are ARE, followed by AVR and AJS. The facility location reformulations obtained the lesser number of feasible solutions being the lowest rank occupied by $AVRFL$, with an overall average of 8 instances per class.

Table 17: Evaluation of the relative gap obtained by the proposed mathematical models.

Classes	Relative Gap (Gap_R) (%)								
($\|T\|/\|P\|/L/h_{it}$)	*AGG-H*	*AJS*	*ARE*	*AVR*	*AGGFL-H*	*AJSFL*	*AREFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S)	68.32	26.82	66.68	55.87	71.9	48	57.62	54.5	62.29
Class 2 (3/10/M)	12.91	6.1	10	17.57	13.63	18.45	10.55	6.65	15.51
Class 3 (3/10/H)	8.8	4.52	5	6.53	7.9	15.1	5.96	3.62	9.41
Mean (3/10)	**30**	**12.48**	**20.23**	**26.65**	**31.14**	**27.18**	**24.61**	**21.59**	**29.07**
Class 4 (3/20/S)	76	45	93.46	75.37	78.14	52.58	71.47	62.22	–
Class 5 (3/20/M)	27.28	23	11.7	22	27.62	22.15	15.28	9.12	22.93
Class 6 (3/20/H)	12	17.61	7	16.75	7.64	20.83	6.92	4.29	14.55
Mean (3/20)	**38.42**	**28.54**	**37.39**	**38**	**37.8**	**31.85**	**31.22**	**25.21**	–
Class 7 (6/10/S)	83.8	55.45	89.85	68.51	83.76	64.2	–	–	–
Class 8 (6/10/M)	18.15	23.27	17.91	18.1	18	23.54	10.79	11.91	14.95
Class 9 (6/10/H)	12.13	19.94	11.15	31.25	12.14	19.93	8.88	9.45	12.8
Mean (6/10)	**38**	**32.87**	**39.28**	**38.44**	**37.96**	**35.89**	–	–	–
Class 10 (6/20/S)	91.59	77	97.51	82.1	69.71	–	–	–	–
Class 11 (6/20/M)	40	51.6	53.63	39.67	15.12	28.22	24.4	18.94	–
Class 12 (6/20/H)	13.4	32.77	21.23	31.65	13.52	25.43	18.48	18.51	5.43
Mean (6/20)	**61**	**53.79**	**57.46**	**54.14**	**32.61**	–	–	–	–
Mean	**41.85**	**31.92**	**38.67**	**44.78**	**34.92**	–	–	–	–

An interesting deduction from the analysis of Table 18 is the impact of instance size on the number of feasible solutions obtained by each model. For instance, the arc-flow models are highly affected by the size of the instance and the length of the items, with Group S always obtaining the smallest number of feasible solutions. Also, doubling the number of items and doubling the number of periods affect the formulations differently. When the number of items is doubled (Class 4), the original formulations obtained fewer feasible solutions than when the number of periods is doubled (Class 7). A different conclusion is drawn for the facility location reformulation, where a fewer number of feasible solutions is attained in Class 7 compared to Class 4.

Table 19 displays, for each formulation and class, the time spent to solve the *MIP*. The $AGG-H$ and $AGGFL-H$ obtained the best solution times. Both formulations spent most of their time in Class 4, while the $AGGFL-H$ solved this class in almost half the time spent by $AGG-H$. As for the models solved exactly, the AJS took the

Table 18: Evaluation of the number of feasible solutions obtained by the proposed mathematical models.

Classes ($\|T\|/\|P\|/L/h_{it}$)	Number of Feasible Solutions								
	AGG-H	*AJS*	*ARE*	*AVR*	*AGGFL-H*	*AJSFL*	*AREFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S)	20	20	20	20	20	20	20	20	12
Class 2 (3/10/M)	20	20	20	20	20	20	20	20	20
Class 3 (3/10/H)	20	20	20	20	20	20	20	20	20
Mean (3/10)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**17.3**
Class 4 (3/20/S)	20	12	20	15	20	2	5	8	0
Class 5 (3/20/M)	20	20	20	17	20	20	20	20	5
Class 6 (3/20/H)	20	20	20	20	20	20	20	20	10
Mean (3/20)	**20**	**17.3**	**20**	**17.3**	**20**	**14**	**15**	**16**	**5**
Class 7 (6/10/S)	20	19	20	20	20	2	0	0	0
Class 8 (6/10/M)	20	20	20	20	20	19	17	19	12
Class 9 (6/10/H)	20	20	20	20	20	20	20	20	14
Mean (6/10)	**20**	**19.67**	**20**	**20**	**20**	**13.67**	**12.33**	**13**	**8.67**
Class 10 (6/20/S)	20	5	17	15	20	0	0	0	0
Class 11 (6/20/M)	20	9	19	19	20	5	5	8	0
Class 12 (6/20/H)	20	18	20	20	20	20	11	18	2
Mean (6/20)	**20**	**10.67**	**18.66**	**18.67**	**20**	**8.33**	**5.33**	**8.67**	**0.67**
Mean	**20**	**16.91**	**19.66**	**19**	**20**	**14**	**13.16**	**14.42**	**7.91**

less amount of time, followed by $AREFL*$ and ARE. The average time almost reached the limit for these models though. We note that for Groups 6/10 and 6/20, the exactly solved models reached the time limit of 1800 seconds for all classes. As for $AGG-H$ and $AGGFL-H$, the results are as follows: $AGG-H$ and $AGGFL-H$ reached the time limit to solve Classes 7 and 10. $AGG-H$ took 143.7 and 60 seconds to solve Classes 8 and 9, respectively, while $AGGFL-H$ spent 73 and 33.5, in this order, for the same classes. $AGG-H$ took 1769 and 1414 seconds to solve Classes 11 and 12, respectively, while $AGGFL-H$ spent 1720 and 1441 seconds, in this order, for the same classes.

Table 19: Evaluation of the *MIP* solution time obtained by the proposed mathematical models for Classes 1 to 6.

Classes ($\|T\|/\|P\|/L/h_{it}$)	Time for the *MIP* solution (seconds) (T_{IP})								
	AGG-H	*AJS*	*ARE*	*AVR*	*AGGFL-H*	*AJSFL*	*AREFL*	*AREFL**	*AVRFL*
Class 1 (3/10/S)	6.44	1800	1800	1800	4.89	1800	1800	1800	1800
Class 2 (3/10/M)	1.34	1497	1612	1719.93	1.66	1800	1593	1546	1719.3
Class 3 (3/10/H)	2.1	1600	1476	1432	1.35	1800	1702.52	1473	1615
Mean (3/10)	**2.26**	**1632.33**	**1629.33**	**1650.64**	**1.96**	**1800**	**1698.5**	**1606.33**	**1711.43**
Class 4 (3/20/S)	1213.43	1800	1800	1800	672.79	1800	1800	1800	–
Class 5 (3/20/M)	96	1800	1800	1800	89.8	1800	1800	1800	1800
Class 6 (3/20/H)	5.52	1600.26	1800	1800	11.46	1800	1800	1800	1800
Mean (3/20)	**438.31**	**1733.42**	**1800**	**1800**	**258**	**1800**	**1800**	**1800**	-
Mean	**220.28**	**1682.87**	**1773**	**1725**	**130**	**1800**	**1746.25**	**1703.16**	–

Even though the overall aspects presented in this section differ from the analysis presented in Section 5.1, where the use of the facility location reformulation resulted in a better performance in aspects such as *MIP* values, absolute and relative gaps, the results obtained from the item length analysis are still encouraging to justify the use of the

extended reformulations in cutting stock problems, both in terms of obtaining stronger lower bounds and also directing the search of the problem with comparable computational performance and better relative gaps.

As reproducibility of data is a crucial concern in the academic environment, the used test instances in this thesis are available for download in https://www.sciencedirect.com/science/article/abs/pii/S0377221722003344?via3Dihub. The seed used to generate the items data set was obtained randomly by the $c++$ random number generator library.

5.3 Lower bounds and object cost

In this section, the models are tested over different object costs in order to study the lower bound sensibility under this parameter. The second set of instances was considered. As concluded by the analysis of Table 14, increasing the instance size only increases the lower bound improvement from the original model to its facility location reformulation, therefore only instances from Groups 3/10 and 3/20 are considered for this analysis, since the overall conclusions regarding these instances will still be valid for larger ones. We also note that since the ARE and AGG models yields the same bounds, the ARE results will be omitted from the analysis. The proposed instances presented convergence issues during the column generation procedure for the $AGGFL$ model, i.e., tailing off effect. Hence, in order to obtain valid lower bounds before the convergence, we use the corresponding Lagrangian bound (see, e.g., Degraeve and Jans (2007) and Vanderbeck and Wolsey (2010)). For this analysis, the $AREFL$ model is considered rather than $AREFL*$ due its lower bound quality.

The time to solve the linear relaxation of all instances varied for each model. On average, the fastest model was AGG with a time, in seconds, of 0.01, followed by $AGGFL$ with 0.02, AJS with 0.063, $AJSFL$ with 0.87, AVR with 0.94 and $AVRFL$ with 1.28 seconds. The ARE and $AREFL$ models took, on average, 27.52 and 48.34 seconds, respectively, to solve the linear problem. An explanation to this behavior can be seen by the features of such classes, in which the length of the items are very small when compared to the objects, hence, the number of possible paths is considerably high, which makes it even difficult to solve the linear relaxations. Throughout the tables, we denote the average lower bound improvement of a class as the average of the improvement from each model in the class.

Table 20 presents for each class and mathematical model, the average values for the LP relaxation when the object cost is equal to 1. Similar to the analysis of Table 14, the facility location reformulation obtains the better lower bounds, which, on average, reach up to 181.68% of improvement for the reformulation of the AJS model, 157.47% when reformulating the ARE model, 153.19% for the reformulation of AVR, and 126% for the reformulation of the AGG model. In general, the facility location lower bound improvement varies according to the item length in the following way, 157% for Classes 1 and 4, 167.75% for Classes 2 and 5, and 139.12% for Classes 3 and 6. We also note that all reformulated models obtained a lower bound greater than the AGG bound.

Table 20: Evaluation of the lower bounds obtained by the proposed mathematical models when $oc = 1$.

Classes	LP relaxation solution (Z_{LP})						
($\|T\|/\|P\|/L/h_{it}$)	AGG	AJS	AVR	$AGGFL$	$AJSFL$	$AREFL$	$AVRFL$
Class 1 (3/10/S)	131.93	107.62	109.45	194.91	278.11	307.19	280.77
Class 2 (3/10/M)	499.44	354.3	487.17	1290.72	1100.68	1381.21	1281.71
Class 3 (3/10/H)	772.75	551.14	768.47	1666.48	1325.57	1770.93	1568.86
Mean (3/10)	**465.26**	**337.68**	**455**	**1050.7**	**901.45**	**1160.43**	**1043.78**
Class 4 (3/20/S)	207.1	156.14	158.73	455.7	518.74	573.65	526.26
Class 5 (3/20/M)	1045.64	741.72	1005.37	2566.19	2151.51	2653.41	2453.48
Class 6 (3/20/H)	1189.72	530.4	727.27	3212.28	1369.72	3277.7	1598.31
Mean (3/20)	**807.37**	**476.1**	**683.36**	**2078.05**	**1346.65**	**2168.25**	**1526**
Mean	**636.31**	**406.89**	**569.18**	**1564.38**	**1124**	**1664.34**	**1284.89**

In Table 21, the lower bounds are computed when considering the object cost equal to 6. In this cost scenario, the reformulation improvement decreases when compared to when $oc = 1$. On average, the facility location reformulation lower bound improvement reach up to 133% for the reformulation of the AJS model, 89.85% for the reformulation of the AVR model, 66.18% when reformulation the ARE model and 52.18% when reformulating the AGG model. In general, the item length groups impact the increase in the lower bounds in the following way, 96.68% for Classes 1 and 4, 82.29% for Classes 2 and 5, and 77.19% for Classes 3 and 6. Note that in Class 3 the AGG average is greater than the $AJSFL$ average. We note that even though the AJS and AVR original and reformulation models attain similar lower bounds for Classes 1 and 4, the AVR and $AVRFL$ models obtain better lower bounds for Classes 2, 3, 5 and 6. We highlight that, on total average, the lower bounds obtained by $AVRFL$ model are 47% better than the $AJSFL$ and AGG lower bounds.

In Table 22, the lower bounds are obtained for the scenario where the object cost is the same as the setup cost, i.e., $oc = sc_t$ ($oc = 100$). On average, the facility location

Table 21: Evaluation of the lower bounds obtained by the proposed mathematical models when $oc = 6$.

Classes	LP relaxation solution (Z_{LP})						
($\|T\|/\|P\|/L/h_{it}$)	AGG	AJS	AVR	$AGGFL$	$AJSFL$	$AREFL$	$AVRFL$
Class 1 (3/10/S)	301.1	155.21	157	360.33	325.9	478.15	325.9
Class 2 (3/10/M)	1240.96	599.3	1164.3	2035.78	1346.26	2125.37	1959.66
Class 3 (3/10/H)	1783.48	806.23	1620.45	2665.99	1643.5	2775.22	2491.63
Mean (3/10)	**1108.51**	**520.25**	**980.58**	**1687.37**	**1105.22**	**1782.56**	**1592.4**
Class 4 (3/20/S)	508.67	205.36	207.57	756.48	574	878.64	574
Class 5 (3/20/M)	2461.51	1030.47	2281.23	3994.56	2444.9	4108.34	3731.87
Class 6 (3/20/H)	2909.36	1195.25	2669.82	4931.76	2918	4998.91	4521.55
Mean (3/20)	**1959.85**	**810.37**	**1719.54**	**3227.6**	**1978.96**	**3312**	**2942.47**
Mean	**1534.18**	**665.31**	**1350**	**2457.48**	**1542**	**2547.28**	**2267.33**

reformulation lower bound improvement reach up to 13.72% for the reformulation of the AJS model, 12.61% for the reformulation of the AVR model, 5.62% when reformulating the ARE model and 4.6% when reformulating the AGG model. For higher object costs, the improvement of the lower bounds from the facility location reformulation decreases. We note that for this cost scenario, the $AVRFL$ obtains smaller lower bounds than the AGG model. In general, the $AVRFL$ model obtains lower bounds that are 250% better than the $AJSFL$ lower bounds and 11% worst than the AGG lower bounds.

Table 22: Evaluation of the lower bounds obtained by the proposed mathematical models when $oc = sc_t$.

Classes	LP relaxation solution (Z_{LP})						
($\|T\|/\|P\|/L/h_{it}$)	AGG	AJS	AVR	$AGGFL$	$AJSFL$	$AREFL$	$AVRFL$
Class 1 (3/10/S)	3480.89	922.7	924.52	3560.21	1089.85	3665.72	1089.84
Class 2 (3/10/M)	14,875.23	5068.36	13,782.66	15,666.58	5148.58	15,751.26	14,577.68
Class 3 (3/10/H)	20,455.33	5844.95	18,133.46	21,343.42	6013.46	21,441.44	19,001.87
Mean (3/10)	**12,937.15**	**3945.34**	**10,946.86**	**13,523.4**	**4083.96**	**13620.71**	**11,556.46**
Class 4 (3/20/S)	6178.58	987.36	989.56	6455.8	1344.94	6554.9	1344.94
Class 5 (3/20/M)	28,739.61	6272.05	26,158.79	30,289	6867.4	30,356.78	27,606.39
Class 6 (3/20/H)	34,711.56	6355.32	31,754.62	36,724.54	7245.91	36,784.08	33,592.1
Mean (3/20)	**23,209.92**	**4538.24**	**19,634.32**	**24,489.78**	**5152.75**	**24,595.61**	**20,847.87**
Mean	**18,073.53**	**4241.79**	**15,290.59**	**19,006.59**	**4618.35**	**19,108.16**	**16,202.13**

In Table 23, the lower bounds are obtained for the scenario where the object cost is the same as the object length, i.e., $oc = L$. On average, the facility location reformulation lower bound improvement is up to 1.59% for the reformulation of the AJS model, 1.41% for the reformulation of the AVR model, 0.58% when reformulating the ARE model and 0.49% when reformulating the AGG model. In general, the $AVRFL$ obtains lower bounds 339% better than the $AJSFL$ bounds and 18.49% worst than the AGG bounds.

As Tables 20, 21, 22 and 23 suggest, the lower bound improvement of the facility

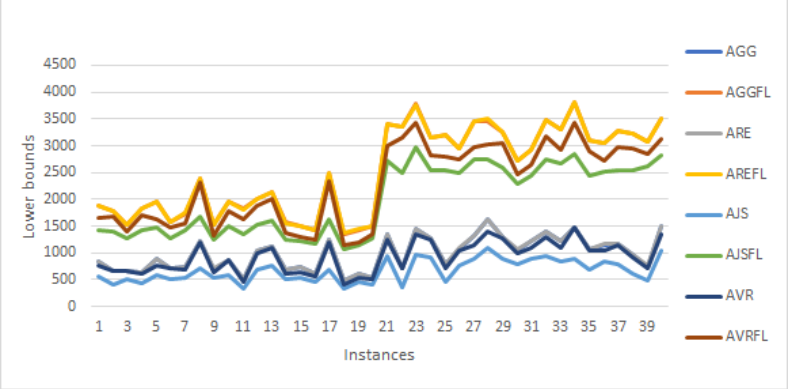
Table 23: Evaluation of the lower bounds obtained by the proposed mathematical models when $oc = L$.

Classes	LP relaxation solution (Z_{LP})						
($\|T\|/\|P\|/L/h_{it}$)	AGG	AJS	AVR	$AGGFL$	$AJSFL$	$AREFL$	$AVRFL$
Class 1 (3/10/S)	33,925.83	8142.52	8142.52	34,030.88	8307.84	34,119.65	8307.84
Class 2 (3/10/M)	145,305.82	40,832.8	134,585.2	146,101	41,193.03	146,183	135,380.3
Class 3 (3/10/H)	199,041.6	47,593.46	176,038.4	199.930.3	47,815	200,031	176,906.8
Mean (3/10)	**126,091.1**	**32,189.59**	**106,255.37**	**126,687.39**	**31,440.62**	**126,791.38**	**106,863.98**
Class 4 (3/20/S)	60,464.97	8258.94	8266.3	60,770.6	8614.31	60,845.7	8614.31
Class 5 (3/20/M)	280,000.75	48,582.71	254,713.19	281,551	49,133.47	281,619.5	256,168.8
Class 6 (3/20/H)	338,892.5	48,015.9	310,124.7	340,894.75	48,373.15	340,966	311,957.1
Mean (3/20)	**226,452.74**	**34,952.52**	**191,034.47**	**227,738.78**	**35,373.64**	**227,788.56**	**192,246.74**
Mean	**176,271.92**	**33,571.1**	**148,645.05**	**177,213.09**	**33,907.13**	**177,289.97**	**149,555.36**

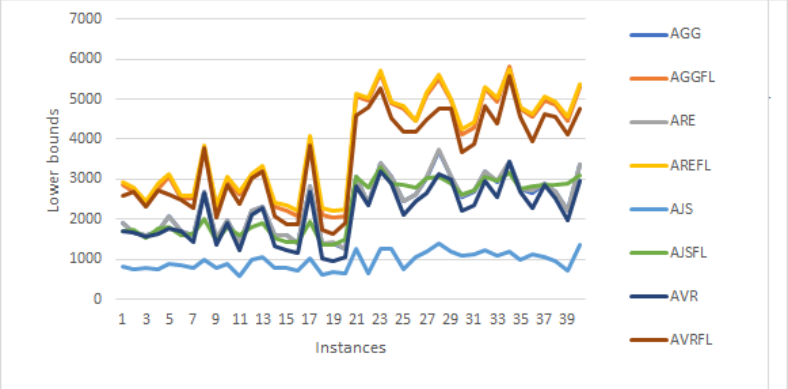
location reformulation over its original formulation is highly affected by the object cost oc. We could also note that in Tables 20 and 21, the average of the LP relaxation value for the $AVRFL$ and $AJSFL$ are greater than the AGG average, however, the statement is not true when considering Tables 22 and 23. The observations mentioned are better depicted in Figure 14, where the plots regarding the instances used in the analysis of Tables 20, 21, 22 and 23, considering Group H, are plotted. We can see that as oc increases, the lower bounds of the original formulation become closer to the lower bounds of the original formulations. A clear distinction between the lower bounds of the original formulations and the facility location reformulations is seen in Figures 14(a) and 14(b), while the curves start to overlap as oc increases. The curves representing the lower bound from AGG and $AVRFL$ start to intersect in Figures 14(c) and 14(d).

Figure 14: Lower bounds obtained by the proposed models considering classes 3/10/H and 3/20/H for different object costs.

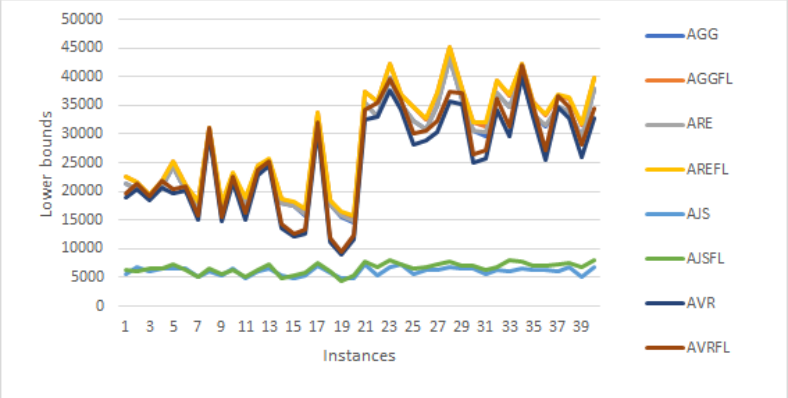
(a) Object cost $oc = 1$.



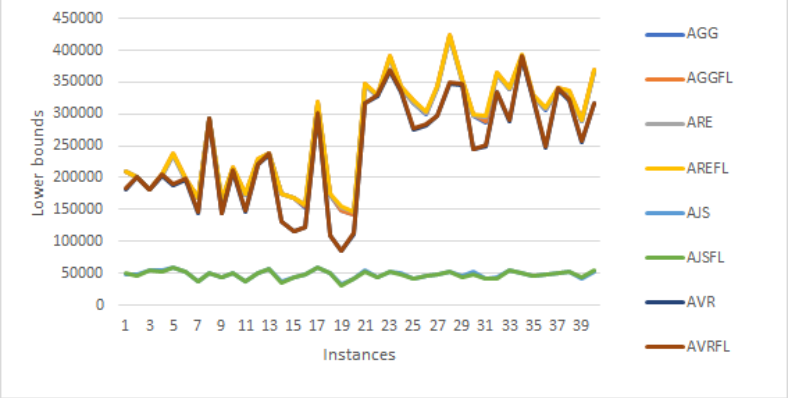
(b) Object cost $oc = 6$.



(c) Object cost $oc = 100$.



(d) Object cost $oc = 1000$.



In order to understand the object cost behavior, we consider the expressions:

$$Z^x_{LP}(F) = oc \sum_{t \in T} \sum_{j \in J} x_{jt} \tag{5.3}$$

$$Z^y_{LP}(F) = \sum_{t \in T} sc_t \sum_{j \in J} y_{jt} \tag{5.4}$$

$$Z^s_{LP}(F) = \sum_{t \in T} \sum_{i \in P} uh_{it} s_{it} \tag{5.5}$$

i. e., the portions of the objective function from a given LP solution regarding variables x, y and s considering formulation F. Without loss of generality, we analyze Expressions (5.3), (5.4) and (5.5) for Class 2 considering ARE and $AREFL$, in order to understand the object cost behavior over the facility location lower bound improvement, and considering AGG, $AJSFL$ and $AVRFL$, in order to understand the lower bound dominance between these models.

Table 24 shows the average values from ARE and $AREFL$ considering the different object costs for Class 2. As we can see, as oc increases, the impact on the column terms is rather limited, the values of $Z^s_{LP}(ARE)$ and $Z^s_{LP}(AREFL)$ tend to increase, the setup cost slightly decreases and the object consumption, $Z^x_{LP}(ARE)/oc$ and $Z^x_{LP}(AREFL)/oc$, tends to equality. Then, as oc increases, the difference, $Z_{LP}(AREFL)$-$Z_{LP}(ARE)$, will present slightly increments while $Z_{LP}(ARE)$ and $Z_{LP}(AREFL)$ will increase due its portions $Z^x_{LP}(ARE)$ and $Z^x_{LP}(AREFL)$. Therefore, the facility location improvement, i.e., $Z_{LP}(AREFL)$-$Z_{LP}(ARE)$, will become relatively small when compared to $Z_{LP}(ARE)$ as oc increases.

Table 24: LP portion average values from ARE and $AREFL$ for Class 2.

LP portion / Costs	$Z^x_{LP}(ARE)/oc$	$Z^y_{LP}(ARE)$	$Z^s_{LP}(ARE)$	$Z^x_{LP}(AREFL)/oc$	$Z^y_{LP}(AREFL)$	$Z^s_{LP}(AREFL)$
$oc = 1$	147	360	0	147.96	982	182.73
$oc = 6$	146.23	358	2.83	146.7	975	260
$oc = sc_t$	144.95	351	29.29	144.97	979	274.64
$oc = L$	144.92	351	33.51	144.92	981	280
mean	145.77	355	16.4	146.14	979	249.34

Now, as regard the lower bound dominance of AGG and $AJSFL$ and $AVRFL$, computational results suggest that when we reformulate F, whether F is the AJS or AVR formulations, the setup constraints tightening obtains a dominance over AGG model in terms of the sums of inventory and setups costs, i.e., $Z^s_{LP}(FFL) \geq Z^s_{LP}(AGG)$ and $Z^y_{LP}(FFL) \geq Z^y_{LP}(AGG)$. However, the linearization techniques incurred by the knapsack constraints still guarantee that $Z^x_{LP}(FFL) \leq Z^x_{LP}(AGG)$. The motivation for this general conclusion is derived from Section 3.6 of the theoretical results.

In Table 25, the average LP portions of $AJSFL$ and $AVRFL$ are displayed for class 2 for illustrative purpose. As we can see, similar behaviors in terms of $AREFL$ model in Table 24 holds for the knapsack based models when c varies. The strength of the $AVRFL$ lower bounds over the $AJSFL$ lower bounds is also highlighted by comparing the $Z^x_{LP}(PFL)/oc$ columns. We note that even though the average of $Z_{LP}(AJSFL)$ is greater than the $Z_{LP}(AGG)$ averages for $oc = 6$ in Class 2 (Table 21), this class already presented five instances that are dominated by $Z_{LP}(AGG)$, while no instances with this property could be found considering the $AVRFL$ lower bounds. For the same class, but now considering $oc = sc_t$, (Table 22), all 20 instances are dominated by the AGG lower bounds when considering $Z_{LP}(AJSFL)$, while only 10 instances satisfied the same for $Z_{LP}(AVRFL)$, and when $oc = L$, 11 instances, considering $Z_{LP}(AVRFL)$, are dominated by $Z_{LP}(AGG)$.

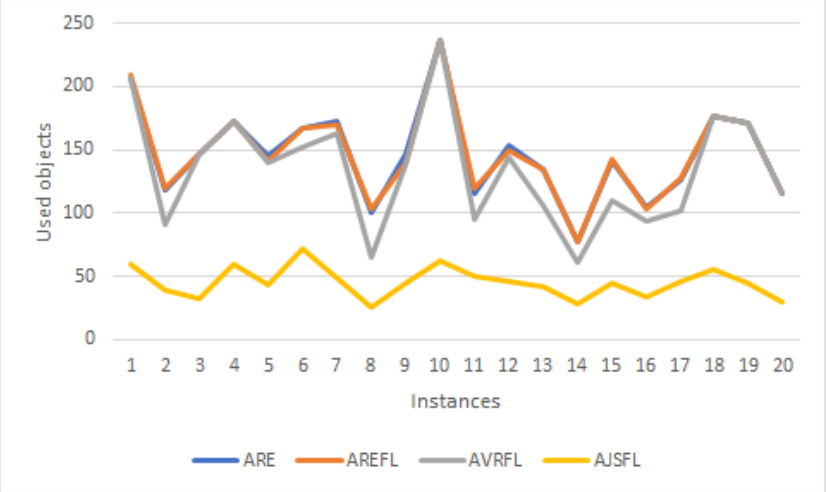
Table 25: LP portion average values from $AJSFL$ and $AVRFL$ for Class 2.

LP portion / Costs	$Z^x_{LP}(AJSFL)/oc$	$Z^y_{LP}(AJSFL)$	$Z^s_{LP}(AJSFL)$	$Z^x_{LP}(AVRFL)/oc$	$Z^y_{LP}(AVRFL)$	$Z^s_{LP}(AVRFL)$
$oc = 1$	52.64	900	149	136.82	961	183.89
$oc = 6$	45.55	895	177.98	134.52	963	189.54
$oc = sc_t$	40.05	873	270.58	134.22	964	191
$oc = L$	40.05	873	320	134.22	964	196.3
mean	44.57	885.25	229.39	134.94	963	190.18

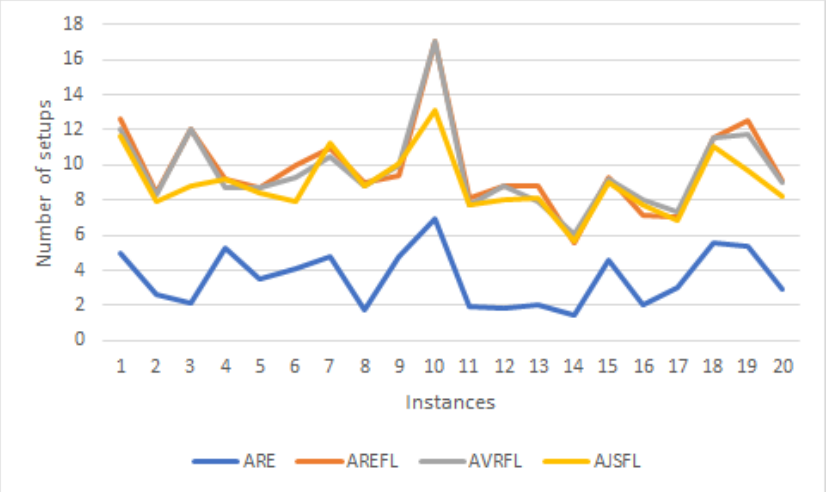
In Figure 15, the number of objects used and the number of setups from ARE, $AREFL$, $AJSFL$ and $AVRFL$ models considering Class 2 is depicted. As the average values in Tables 24 and 25 suggest, the objects used considering ARE, $AREFL$ and $AVRFL$ models overlap each other, with the arc-flow models obtaining the highest values. The $AJSFL$ model obtains the lowest number of used objects for all cost scenarios. This behavior is not strict for the cost scenarios of Figures 15(a), 15(b), 15(c) and 15(d) and similar behaviors are observed for different object costs, oc.

Figure 15: Number of objects (right) and setups (left) from *ARE*, *AREFL*, *AJSFL* and *AVRFL* for Class 2.

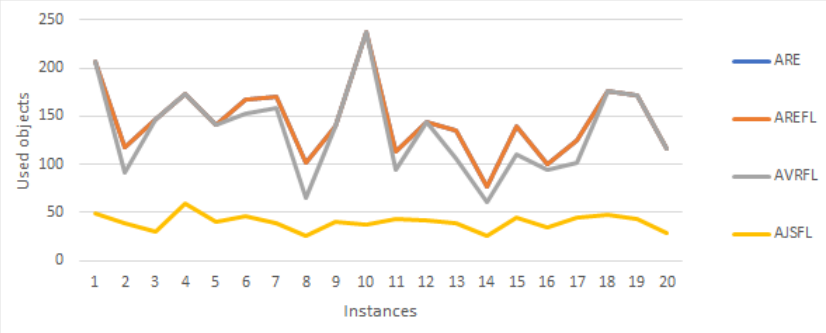
(a) Number of objects for $oc = 6$.



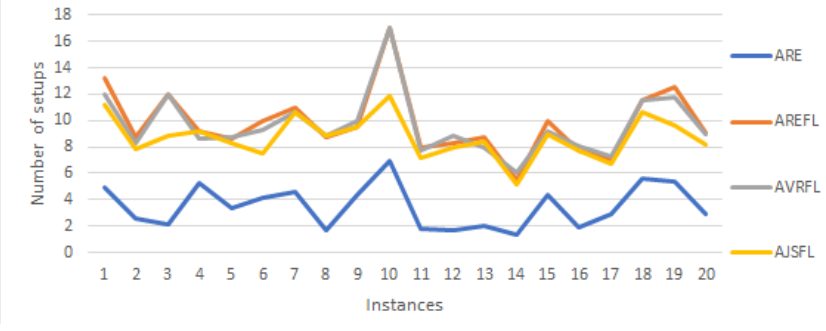
(b) Number of setups for $oc = 6$.



(c) Number of objects for $oc = 1000$.



(d) Number of setups for $oc = 1000$.



5.4 Computational experiments for the ABRKGA

In order to evaluate D_1 (Subsection 4.2.3.2) and D_2 (Subsection 4.2.3.3), we consider the second set of instances and the $AGG - H$ solution method. A machine with Intel Core i7 3.2 GHz CPU processor and 16 GB of RAM was used to conduct the tests. In this experiment, the methods were implemented and solved using Visual Studio 2015 and the C callable library of IBM ILOG Cplex 22.1.

As for the object cost, oc, a number of papers vary in the way it is considered regarding the setup cost, sc_t. In Moretti and de Salles Neto (2008), $oc = sc_t/5$ and $oc = sc_t/10$ are considered. Golfeto et al. (2009b) considered, $oc = sc_t$, $oc = sc_t/5$ and $oc = sc_t/10$. In Mobasher and Ekici (2013), the object length (L) is taken into account to compute the object cost. The total material cost is given by Loc. Hence, considering the values of L from the test instances used by the authors, we provide only an estimation of the relationships between the object and setup costs. On general, the costs varied from $oc \approx 90sc_t$ down to $oc \approx sc_t/10$ in four cost scenarios. In Cui et al. (2015), two scenarios where $oc = 10c_t$ and $oc = sc_t$ are considered. Therefore, in order to evaluate the robustness of the encoders and respective decoders as oc changes, we considered three cost scenarios such that $oc = sc_t/10$, $oc = sc_t$ and $oc = 10sc_t$ ($oc = L$).

The lower bounds, Z_{LP} (obtained from $AREFL$), the upper bounds (fitness function), Z_{IP}, and the time to obtain the solution, T_{IP}, are used for comparison. The analysis is performed in terms of item length groups rather than groups classified according to the number of periods and items. As with the previously sections, the time limit is set as 1800 seconds. The initial size of the $ABRKGA$ population, p, is set to 500 and the parameter λ is set to 0.999. These values were determined through preliminary computational tests and achieved the best ratio between integer solutions and run time considering the proposed decoders when different values of p and λ were considered. Since $ABRKGA$ is a stochastic approach, the number of runs for each instance was considered as 15. We consider, throughout the next tables, the values obtained from the best run of the $ABRKGA$ for comparison.

In order to understand the impact of the $ABRKGA$ in terms of item length, we separate the instances according to item length groups, rather than per instance size as in the previous sections.

In Table 26 the average results when oc is 10, i.e. the object cost is ten times smaller than the setup cost, are displayed. For this scenario, D_1 presented the best solution times

and the best integer values except for the integer values from Class 11, where $AGG-H$ obtained an improvement of 0.9%, respectively, and for instances in Group S, where D_2 obtained an average improvement of 7%. The $ABRKGA$ decoders presented their best performance when compared to $AGG-H$, for both integer values and solution times, in instances from group S. As for groups M and H, similar averages are observed for the proposed heuristics. In general, D_1 obtained the best solution averages and solution times. D_2 obtained a feasible solution average smaller than $AGG-H$ (2.68%) in 94.37% less time than $AGG-H$ and 92.68% more time than D_1. The absolute gaps for the $AGG-H$ are, on average, 56.62% for Group S, 10.58% for Group M and 7.97% for Group H.

Table 26: Evaluation of the lower bounds, upper bounds (feasible solutions) and solution time obtained by the proposed heuristic procedures when $oc = 10$.

Class	lower bound	solution (Z_{IP})			solution (T_{IP})		
($\|T\|/\|P\|/l_i$)	Z_{LP}	D_1	D_2	$AGG-H$	D_1	D_2	$AGG-H$
Class 1 (3/10/S)	613.97	892.48	891.56	1421.41	0.28	3.64	9.83
Class 4 (3/20/S)	1121.28	1558.00	1516.44	2861.51	0.69	9.41	991.26
Class 7 (6/10/S)	1188.54	2006.11	1857.5	2512.00	0.68	10.5	1689.00
Class 10 (6/20/S)	2413.39	4083.94	3679.99	5501.12	1.31	20	1764.78
Mean (S)	**1334.29**	**2135.13**	**1993.87**	**3074.00**	**0.74**	**10.89**	**1113.72**
Class 2 (3/10/M)	2695.72	2872.69	2925.52	3022.73	0.31	13.75	0.24
Class 5 (3/20/M)	5187.74	5494.76	5660.32	5762.9	1.04	37.43	88
Class 8 (6/10/M)	5638.51	6055.15	6472.21	6258.7	0.98	35.32	100.5
Class 11 (6/20/M)	9737.59	10,865.63	11,730.09	10,767.5	2.12	66.57	1524.85
Mean (M)	**5314.84**	**6322.06**	**6697.03**	**6452.96**	**1.11**	**38.27**	**428.4**
Class 3 (3/10/H)	3567.99	3762.85	3817.98	3883.49	0.77	20.14	0.1
Class 6 (3/20/H)	6352.66	6645.76	6881.39	6888.89	5.35	51.11	14.17
Class 9 (6/10/H)	6554.83	6918.18	7391.27	7057.16	4.2	41.1	8.73
Class 12 (6/20/H)	12,597.48	13,340.6	14,941.03	13,651.39	11.64	93.15	956.3
Mean (H)	**7268.24**	**7666.85**	**8258.17**	**7870.23**	**5.49**	**51.3**	**244.82**
Mean	**4639.12**	**5374.68**	**5649.69**	**5799.06**	**2.45**	**33.49**	**595.64**

Table 27 shows the average results for when oc is equal the setup cost, sc_t. The decoders outperformed the $AGG-H$ in both computational time and feasible solution average for instances from Group S, where D_2 obtained a feasible solution average 4.16% smaller than D_1 average, but spent 94.39% more time than D_1 to solve the instances. The $AGG-H$ outperformed both D_1 and D_2 for all classes from groups M and H. In general, D_1 and D_2 obtained integer solutions 1.06% and 1.37% higher, respectively, than the $AGG-H$ solutions. While D_2 spent 93.89% less time than $AGG-H$, D_1 took 99.6% less time than $AGG-H$ to solve all instances. The absolute gaps for the $AGG-H$ are, on average, 20.89% for Group S, 2.17% for Group M and 1.61% for Group H.

Table 28 displays the average results for when the object cost is ten times the setup cost. For this scenario, D_1 was outperformed by $AGG-H$ for all classes, while D_2

Table 27: Evaluation of the lower bounds, upper bounds (feasible solutions) and solution time obtained by the proposed heuristic procedures when $oc = sc_t$.

Class	lower bound	solution (Z_{IP})			solution (T_{IP})		
($\|T\|/\|P\|/l_i$)	Z_{LP}	D_1	D_2	$AGG-H$	D_1	D_2	$AGG-H$
Class 1 (3/10/S)	3665.72	4175.9	4120.31	4653.64	0.33	5.17	196.7
Class 4 (3/20/S)	6554.9	7468.14	7216.53	8685	0.74	13.31	1484.39
Class 7 (6/10/S)	7206.48	8673.15	8335.43	8703.95	0.6	11.38	1721.04
Class 10 (6/20/S)	14,504.96	17,699.16	16,825.63	18,055.66	1.25	22.21	1741.68
Mean (S)	**7983**	**9504.01**	**9124.47**	**10,024.56**	**0.73**	**13.02**	**1285.95**
Class 2 (3/10/M)	15,751.26	16,076.28	16,128.11	16,109.2	0.37	16.41	0.25
Class 5 (3/20/M)	30,356.78	31,262.24	31,279.12	31,024.73	1.18	44.98	177.11
Class 8 (6/10/M)	34,098.52	35,065.49	35,300.94	34,724.72	1.05	39.2	111.09
Class 11 (6/20/M)	57,750.06	61,830.19	61,599.7	58,809	1.96	72.52	1406
Mean (M)	**34,489.14**	**36,058.54**	**36,076.97**	**35,166.91**	**1.14**	**43.28**	**423.61**
Class 3 (3/10/H)	21,441.44	21,845.64	21,886.52	21,840.85	1.35	24.67	0.08
Class 6 (3/20/H)	36,784.08	37,538.16	37,848.44	37,387.56	6.12	65.3	7.42
Class 9 (6/10/H)	38,917.22	39,743.84	40,275.61	39,413.38	4.93	48.32	7.11
Class 12 (6/20/H)	74,761.72	77,643.64	79,277.98	75,828.28	11.06	112.33	935.4
Mean (H)	**42,976.11**	**44,192.82**	**44,822.14**	**43,614.52**	**5.86**	**62.65**	**237.5**
Mean	**28,482.75**	**29,918.29**	**30,007.86**	**29,602**	**2.58**	**39.65**	**649**

outperformed $AGG-H$ only in Class 4. In general, D_1 presented similar values except for Class 11, where D_2 obtained a 0.64% better solution value. The increment between D_2 integer averages and $AGG-H$ averages are as follows: -0.85% for Group S, 0.81% for Group M and 0.69% for Group H. In general, D_2 spent 95.41% less time than $AGG-H$ and 94.26% more time than D_1 to solve all instances. The absolute gaps for the $AGG-H$ are, on average, 4.54% for Group S, 0.36% for Group M and 0.34% for Group H.

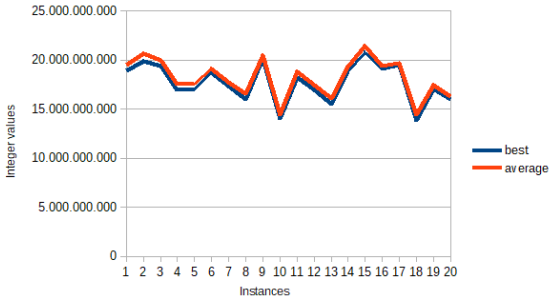
Table 28: Evaluation of the lower bounds, upper bounds (feasible solutions) and solution time obtained by the proposed heuristic procedures when $oc = 10sc_t$.

Class	lower bounds	solution (Z_{IP})			solution (T_{IP})		
($\|T\|/\|P\|/l_i$)	Z_{LP}	D_1	D_2	$AGG-H$	D_1	D_2	$AGG-H$
Class 1 (3/10/S)	34,119.65	35,954.7	35,676.28	35,767.35	0.32	6.65	958.25
Class 4 (3/20/S)	60,845.7	65,346.29	62,811.63	65,288.45	0.72	17.17	1787.66
Class 7 (6/10/S)	67,275.45	70,611.8	69,646.21	69,244.41	0.61	13.79	1619.83
Class 10 (6/20/S)	135,322.5	144,829.68	139,767.34	140,241.3	1.25	32.57	1800
Mean (S)	**74,390.32**	**79,185,62**	**76,975.35**	**77,635.38**	**0.72**	**17.54**	**1541.43**
Class 2 (3/10/M)	146,183	147,580.13	147,962.45	146,839	0.39	19.27	221
Class 5 (3/20/M)	281,619.5	285,635.1	284,364.86	282,533	1.14	52.95	711.29
Class 8 (6/10/M)	318,356.6	320,268.45	320,380.07	319,238	1.00	42.29	760.59
Class 11 (6/20/M)	536,818.05	547,918.65	544,393.42	538,106.8	1.8	71.93	935.4
Mean (M)	**320,744.29**	**325,350.58**	**324,275.2**	**321,679.2**	**1.08**	**46.61**	**657**
Class 3 (3/10/H)	200,031	201,944.44	201,800.05	201,036.7	1.5	28.20	94.38
Class 6 (3/20/H)	340,966	344,810.14	344,806.76	342,039.2	6.35	76.26	614
Class 9 (6/10/H)	361,556	363,668.75	363,819.51	362,334.6	4.81	51.4	580.12
Class 12 (6/20/H)	695,526	702,402.96	703,336.5	697,206.2	10.44	114.51	1399.43
Mean (H)	**399,519.75**	**403,206.57**	**403,440.7**	**400,654.17**	**5.77**	**67.59**	**671.98**
Mean	**264,884.79**	**269,247.59**	**269,230.42**	**266,656.25**	**2.52**	**43.91**	**956.8**

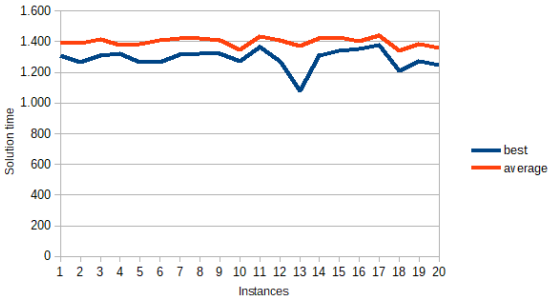
Figure 16 presents the best and average solution values and times for Class 6/20 considering D_1 with 15 runs and $oc = c_t$. A similar analysis is made regarding the solution values and times. The best and average values present slight variations from each other, while the biggest variations happen in Group S of small items. We highlight that the same behavior in the plots of Figure 16 is found when D_2 is considered (though with different values) and when different classes of instances and other object costs scenarios are considered.

Figure 16: Best and average solution values (right) and time (left) for Class 6/20 considering D_1 with 15 runs.

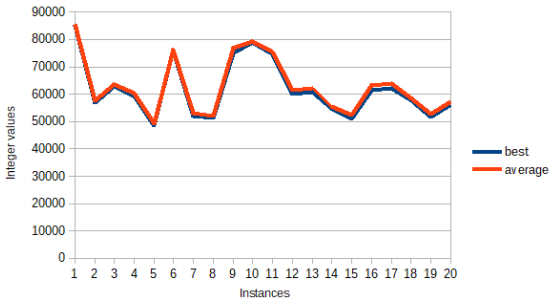
(a) Group S solution values.



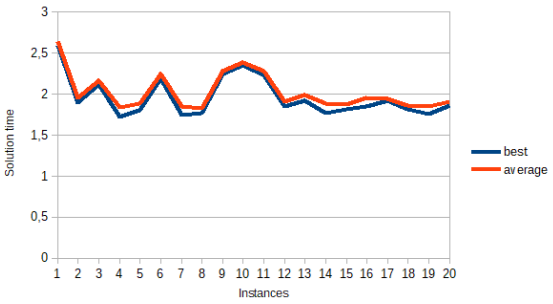
(b) Group S solution times.



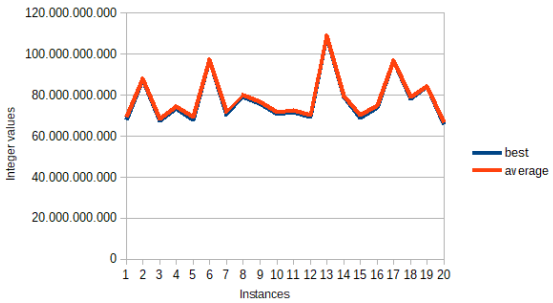
(c) Group M solution values.



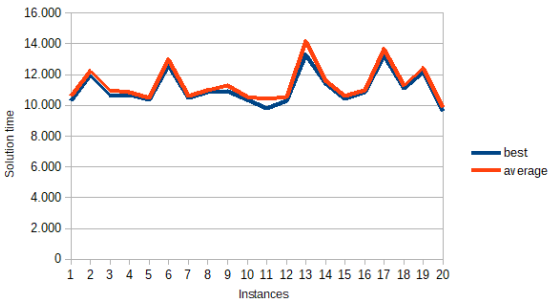
(d) Group M solution times.



(e) Group H solution values.



(f) Group H solution times.



Throughout Tables 26, 27 and 28, we note that D_1 is generally better than D_2 for costs $oc = 10$ and $oc = sc_t$, while D_2 presents better averages than D_1 only when $oc = L$. Better solution values than D_1 are obtained from D_2 for all instances from Group S. D_1 also dominated the solution time through all instances compared to D_2 and the column

generation heuristic.Next, due the performance of the proposed decoders for scenarios where object cost is equal or higher than the setup cost, we conduct an experiment in order to compare the cutting patterns obtained via column generation as descried in Chapter 5 and the cutting patterns obtained from the $ABRKGA$ population after convergence.

5.4.1 Additional experiment

The idea consists as follows: we consider a limited set $\overline{J}$ of cutting patterns from the $ABRKGA$ solutions, then after initializing the master AGG problem with the homogeneous cutting patterns, instead of solving the knapsack subproblem (4.1)-(4.3) for each t, the cutting pattern with most reduced cost from $\overline{J}$ is add to the AGG master problem in period t. When no improvement in the LP solution of the master problem is noticed, the procedure stops and the integer problem is solved. The linear and integer problems are solved by the optimization software.

The set $\overline{J}$ is composed with the cutting patterns from the best 200 individuals of the final generation of the $ABRKGA$. The experiment was conducted using D_1 rather than D_2 due its better computational performance. Just as with the previous results, the total time solution is set to be 1800. The comparison is made using integer values and solution times. We denote the combined procedure as $D_1 + S$ throughout the solution tables.

Table 29 presents the results for $oc = sc_t$. For this case, the combined heuristic slightly improves the integer solution for all classes when compared to $AGG - H$ (1% better considering all classes). As for the computational time, $D_1 + S$ was faster only in Group S and classes 5, 8, 11 and 12. In general, for each group, $D_1 + S$ was 46.22% faster when solving instances from Group S, 11.44% faster in Group M and 6.13% faster in Group H. In terms of total time average, $D_1 + S$ was 33.76% faster.

Table 30 displays the comparison for $oc = 10sc_t$. For this case, $AGG - H$ obtained better integer solution averages for all groups with improvements of 2.46% for Group S, 0.32% for Group M and 0.09% for Group H. We note that even though the $D_1 + S$ procedure obtained better solution values for all groups when comparing to D_1 alone (Table 28), the decoder cutting patterns could not improve the integer results for Group S as it did to Groups M and H. As for computational time, the combined heuristic obtained better averages for all classes. In general, $D_1 + S$ was 61.08% faster than $AGG - H$ and obtained 0.42% higher feasible solutions.

We conclude by the results shown in tables 29 and 30 that the cutting patterns

Table 29: Integer solutions and solution times when considering $D_1 + S$ and $AGG - H$ when $oc = sc_t$.

Class	solution (Z_{IP})		solution (T_{IP})	
($\|T\|/\|P\|/l_i$)	$D_1 + S$	$AGG - H$	$D_1 + S$	$AGG - H$
Class 1 (3/10/S)	4267.3	4653.64	45.92	196.7
Class 4 (3/20/S)	7678.4	8685	360.00	1484.39
Class 7 (6/10/S)	8413.23	8703.95	1370.5	1721.04
Class 10 (6/20/S)	16,997.68	18,055.66	989.79	1741.68
Mean (S)	**9339.15**	**10,024.56**	**691.55**	**1285.95**
Class 2 (3/10/M)	16,051	16,109.2	0.9	0.25
Class 5 (3/20/M)	31,015.8	31,024.73	135.5	177.11
Class 8 (6/10/M)	34,550.81	34,724.72	76.62	111.09
Class 11 (6/20/M)	58,702.07	58,809	1287.49	1406
Mean (M)	**35,079.92**	**35,166.91**	**375.13**	**423.61**
Class 3 (3/10/H)	21,718.94	21,840.5	1.29	0.08
Class 6 (3/20/H)	37,359.6	37,387.56	7.54	7.42
Class 9 (6/10/H)	39,308.19	39,413.38	22.8	7.11
Class 12 (6/20/H)	75,562.26	75,828.28	860.08	935.4
Mean (H)	**43,487.25**	**43,614.52**	**222.93**	**237.5**
Mean	**29,302.1**	**29,602**	**429.87**	**649**

Table 30: Integer solutions and solution times when considering $D_1 + S$ and $AGG - H$ when $oc = 10sc_t$.

Class	solution (Z_{IP})		solution (T_{IP})	
($\|T\|/\|P\|/l_i$)	$D_1 + S$	$AGG - H$	$D_1 + S$	$AGG - H$
Class 1 (3/10/S)	36,121.09	35,767.35	583.00	958.25
Class 4 (3/20/S)	66,922.48	65,288.45	178	1787.66
Class 7 (6/10/S)	70,110.84	69,244.41	930.1	1619.83
Class 10 (6/20/S)	145,234.61	140,241.3	651.58	1800
Mean (S)	**79,597.25**	**77,635.38**	**585.67**	**1541.43**
Class 2 (3/10/M)	147,127	146,839	66.85	221
Class 5 (3/20/M)	284,319.56	282,533	189.00	711.29
Class 8 (6/10/M)	319,590.79	319,238	301.56	760.59
Class 11 (6/20/M)	539,877.39	538,106.8	713.4	935.4
Mean (M)	**322,728.68**	**321,679.2**	**317.7**	**657**
Class 3 (3/10/H)	200,765.79	201,036.7	2.78	94.38
Class 6 (3/20/H)	343,535.29	342,039.2	250.22	614
Class 9 (6/10/H)	362.355.63	362,334.6	15.82	580.12
Class 12 (6/20/H)	697,481.31	697,206.2	585.47	1399.43
Mean (H)	**401,034.5**	**400,654.17**	**213.57**	**671.98**
Mean	**267,786.81**	**266,656.25**	**372.31**	**956.8**

generated by the *ABRKGA* are attractive columns from both the column generation technique and the integer problem solution.

CHAPTER 6

CONCLUSION AND FUTURE RESEARCHES

The multi-period cutting stock problem (*MPCSP*) is studied in this thesis. This problem addresses multiple periods in a finite planning horizon, the inventory of items which comprises the link between periods, and the cutting process of objects in each period. In this way, the decision-making of such processes might occur at different levels of the supply chain. For instance, the planning managers are usually responsible for the production planning of the items in order to meet the demand, whereas the machine manufacturers perform the optimization of the cuts in the cutting process. In this work, we proposed eight formulations for the *MPCSPs*, one formulation was adapted from the literature, while seven others were new formulations proposed for the problem, four of which were reformulations based on the facility location problem with stronger lower bounds. A thorough theoretical study regarding lower bound strength was conducted in order to establish a comparative understanding among these formulations. Except for the lower bound relationship between the *AGG* model and the knapsack based facility location reformulations, a dominance relation could be found for all other models. In addition, a computational study was performed based on randomly generated instances to evaluate the formulations in terms of computational performance and so that theoretical relationships could be better understood.

The computational experiments were performed over two sets of instances. In the first set, we fix the item length and vary the holding costs and the object length while in the second, we fix the holding costs and the object length and vary the item length, resulting in more difficult instances. For both sets of instances the integer solutions were obtained considering the setup cost as 100 times the object cost. Regarding the upper bound

achievements of the proposed formulations when using the first data set, the $AGGFL$ and $AREFL^*$ models were able to find the smallest relative gaps on 22 out of 24 classes, with 1.77% and 2.11% on average, respectively. In addition, the $AGGFL$ and $AREFL^*$ models were in average always faster than their respective original formulations. As for the second set of instances, the pattern based formulations present slight differences on overall integer solution while $AGGFL$ showed a better computational performance. The AJS model obtained the best average of relative gaps for classes with small items, however, as the instances size increase, the model could not find all feasible solutions, whereas AVR could find more feasible solutions than AJS for all classes (though with bigger relative gaps). The arc-flow formulations obtained, in general, the best relative gaps for instances with medium and high item lengths, whereas $AREFL^*$ obtained the best gaps for classes where feasible solutions could be found. In general, as the item length becomes smaller in relation to the object length the facility location formulation presented more difficulties in finding feasible solution for the problem. This observation is clearer when the knapsack based models were considered, since their facility location reformulation had difficulties even for obtaining smaller relative gaps than the respective original formulations. In general, except for the pattern-based formulation, the arc-flow and knapsack-based models struggle to solve medium-size instances where 6 periods and 20 items are considered. As for the lower bounds, the facility location improvement is highly affected by the increase of the object cost value. However, all experiments showed that the facility location reformulations improve the quality of the lower bounds. Considering the second set of instances, the improvement ranges from 139% to 181%, when the object cost is considered as 1, down to 0.49% to 1.59% when the cost is considered as 1000. We highlight that Chapters $1, 2, 3$ and 4 were partially published in Silva et al. (2023).

A metaheuristic based on the Adaptive Biased Random Key Genetic Algorithm ($ABRKGA$) is also presented for the problem. A chromosome represented as random item production in different periods and rules to construct cutting patterns where a decoder procedure uses these information to obtain a feasible cutting plan is presented. Two different processes of encoding and decoding, D_1 and D_2, were proposed based on different ways of selecting the production and construct the cutting patterns for each period. A column generation approach based on the AGG model is also used for comparative purpose. The solution procedures where tested using the second set of instances. Three costs Scenarios such that the setup price is 10 times bigger, equal and 10 times smaller than the object cost, denoted here as Scenarios 1, 2 and 3, respectively, were considered for test evaluation.

The computational experiments show that D_1 and D_2 outperforms the column generation technique for all instances in the classes with small items with cost Scenarios 1 and 2. D_2 is the only decoder which obtained comparable results to the column generation procedure for the classes with small items when considering the third Scenario. Comparable results for the medium and high item length classes were also found for Scenarios 1 and 2 while the column generation outperforms both decoders in these classes when Scenario 3 is considered. D_1 is almost 3 times faster than D_2 and slight variation in computational time is noted from both decoders as the object cost varies. In general, D_1 works better than D_2 for smaller object costs while D_2 works better than D_1 when higher costs were considered. An additional experiment using D_1 and the column generation procedure is conducted for the costs Scenarios 2 and 3. The combined procedure obtained comparable results in all instances in better computational time for both costs configuration.

An interesting subject for future research is to extend the theoretical insights and reformulations proposed in this thesis to cutting stock problems considering different practical aspects, such as: several machines, capacity constraints, sequence-dependent cut setups (Arbib and Marinelli (2007) and Wuttke and Heese (2018)), among others. In addition, other strength strategies from the lot-sizing literature, such as shortest path reformulation and facet defining inequalities can be extended to enhance formulations based on cutting stock problems. The column-and-row method addressed by Sadykov and Vanderbeck (2013) may also be used to generate cutting patterns considering the facility location reformulation. Considering the capacitated case, applying Lagrangian relaxation to the capacity constraint will result in a sub-problem similar to the one studied in this thesis. A multi-objective approach can also be applied to analyze the complex trade-offs present in the objective function.

As for future research regarding the *ABRKGA*, sequential heuristics such as the one proposed by Haessler (1975) may be adapted in the decoder since such strategies have been proven effective to deal with items with high length. Other strategies such as implementing local search and adapting the decoder to consider capacity constraints are interesting topics of research. In addition, the work of Chaves and Lorena (2021) proposed a *BRKGA* with Q-Learning algorithm ($BRKGA - QL$), where the *BRKGA* parameters are set during the evolutionary process using Reinforcement Learning, indicating the best configuration at each stage. Therefore, testing decoders under the $BRKGA - QL$ framework is also a research topic.

We also would like to highlight that the feature considered in this thesis, where a cutting pattern produces several items, can be extended to several process industries,

where the products are obtained by processes that can produce several types of products simultaneously. These processes can be a specific mode of configuration of a production system that can produce several different items simultaneously and in varied quantities. A recent general discussion about it can be found in Villas Boas et al. (2021). Examples of such process industries are: refineries (Göthe-Lundgren et al. (2002) and Shi et al. (2014)), molded pulp (Martínez et al. (2018, 2019)), electrofused grains (Luche et al. (2009)), foundry (de Araujo et al. (2008)), offset printing industry (Baumann et al. (2015)) industries, among others. In practice, industries define a list of alternative processes as input data, and the decision is related to the selection of the processes to be used in each period of the planning horizon. However, according to Martínez et al. (2019), for some production environments, the number of configurations might be large and hence the complete enumeration not possible while considering only a subset of them may lead to sub-optimal solutions. Hence, an integrated approach that considers the process configuration together with other decisions is also an interesting avenue for future research.

REFERENCES

Absi, N., Dauzère-Pérès, S., Kedad-Sidhoum, S., B., P., & Rapine, C. (2016). The single-item green lot-sizing problem with fixed carbon emissions. *European Journal of Operational Research*, *248*(3), 849–855.

Absi, N., & van den Heuvel, W. (2019). Worst case analysis of relax and fix heuristics for lot-sizing problems. *European Journal of Operational Research*, *279*(2), 449–458.

Agrawal, S., Subramanian, KR., & Kapoor, S. (2010). Operations research - contemporany role in managerial decisions. *Internation Journal of Recent Researches and Aplied Studies (IJRRAS)*, *3*(2), 200–208.

Akartunali, K., Fragkos, I., Miller, A. J., & Wu, T. (2016). Local cuts and two-period convex hull closures for big-bucket lot-sizing problems. *INFORMS J Comput.*, *28*(4), 766–780.

Aktin, T., & Özdemir, R. G. (2009). An integrated approach to the one-dimensional cutting stock problem in coronary stent manufacturing. *European Journal of Operational Research*, *196*(2), 737–743.

Alem, D., & Morabito, R. (2013). Risk-averse two-stage stochastic programs in furniture plants. *OR Spectrum*, *35*(4), 773–806.

Alem, D., Oliveira, J. F., & Peinado, M.C.R. (2020). A practical assessment of risk-averse approaches in production lot-sizing problems. *International Journal of Production Research*, *58*, 2581–2603.

Aloisio, A., Arbib, C., & Marinelli, F. (2011a). Cutting stock with no three parts per pattern: Work-in-process and pattern minimization. *Discrete Optimization*, *8*(2), 315–332.

Aloisio, A., Arbib, C., & Marinelli, F. (2011b). On lp relaxations for the pattern minimization problem. *Networks*, *57*(3), 247–253.

Alves, C., Clautiaux, F., Valério de Carvalho, J. M., & Rietz, J. (2016). *Dual-feasible functions for integer programming and combinatorial optimization*. EURO Advanced Tutorials on Operational Research (Springer International Publishing).

Alves, C., & Valério de Carvalho, J. M. (2008a). A branch-and-price-and-cut algorithm for the pattern minimization problem. *RAIRO- Operations Research*, *42*, 435–453.

Alves, C., & Valério de Carvalho, J. M. (2008b). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, *35*(4), 1315–1328.

Arbib, C., & Marinelli, F. (2005). Integrating process optimization and inventory planning in cutting-stock with skiving option: An optimization model and its application. *European Journal of Operational Research*, *163*(3), 617–630.

Arbib, C., & Marinelli, F. (2014). On cutting stock with due dates. *Omega*, *46*, 11–20.

Arbib, C., & Marinelli, F. (2007). An optimization model for trim loss minimization in an automotive glass plant. *European Journal of Operational Research*, *183*(3), 1421–1432.

Attila, Ö. N., Agra, A., Akartunali, K., & Arulselvan, A. (2021). Robust formulations for economic lot-sizing problem with remanufacturing. *European Journal of Operational Research*, *288*(2), 496–510.

Barany, I., Van Roy, T.J., & Wolsey, L.A. (1984). Strong formulations for multi-item capacitated lot-sizing. *Manag. Sci.*, *30*(10), 1255–1261.

Baumann, P., Forrer, S., & Trautmann, N. (2015). Planning of a make-to-order production process in the printing industry. *Flex. Serv. Manuf. J.*, *27*, 534–560.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, *6*(2), 154–160.

Ben Amor, H. (1997). *Résolution du problème de découpe par génération de colonnes* [Doctoral dissertation, École Polytechnique de Montréal].

Ben Amor, H., & Valério de Carvalho, J. M. (2005). Cutting stock problems. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 131–161). Springer US.

Blickle, T., & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, *4*(4), 361–394.

Bonnevay, S., Gavin, G., & Aubertin, P. (2016). Comparison of two metaheuristics to solve a 2-d cutting stock problem with set-up cost in the paper industry. *Int. J. Metaheuristics*, *5*(1), 31–50.

Braga, N., Alvez, C., Macedo, R., & Valério de Carvalho, J. M. (2015). A model-based heuristic for the combined cutting stock and scheduling problem. *Lecture Notes in Computer Science*, *9156*, 409–505.

Brahimi, N., Absi, N., Dauzère-Pérès, S., & Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, *263*(3), 838–863.

Chaves, A., Gonçalves, J. F., & Lorena, L. (2018). Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers & Industrial Engineering*, *124*, 331–346.

Chaves, A., & Lorena, L. H. N. (2021). *An adaptive and near parameter-free brkga using reinforcement learning* (tech. rep.). Universidade Federal de São Paulo (UNIFESP).

Christofoletti, M. M., de Araujo, S. A., & Cherri, A. C. (2021). Integrated lot-sizing and cutting stock problem applied to the mattress industry. *Journal of the Operational Research Society*, *72*(6), 1279–1293.

Côté, J. F., & Iori, M. (2018). The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, *30*(4), 646–661.

Cui, Y., Yang, L., Zhao, Z., Tang, T., & Yin, M. (2013). Sequential grouping heuristic for the two-dimensional cutting stock problem with pattern reduction. *International Journal of Production Economics*, *144*(2), 432–439.

Cui, Y., Zhong, C., & Yao, Y. (2015). Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. *European Journal of Operational Research*, *243*(2), 540–546.

de Araujo, S. A., Arenales, M. N., & Clark, A. R. (2008). Lot sizing and furnace scheduling in small foundries [Part Special Issue: New Trends in Locational Analysis]. *Computers & Operations Research*, *35*(3), 916–932.

de Araujo, S. A., de Reyck, B., Degraeve, Z., Fragkos, I., & Jans, R. (2015). Period decompositions for the capacitated lot sizing problem with setup times. *INFORMS Journal on Computing*, *27*(3), 431–448.

de Araujo, S. A., Poldi, K. C., & Smith, J. (2014). A genetic algorithm for the one-dimensional cutting stock problem with setups. *Pesquisa Operacional*, *34*, 165–187.

Degraeve, Z., & Jans, R. (2007). A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, *55*(5), 909–920.

Dekker, R., Bloemhof, J., & Mallidis, I. (2012). Operations research for green logistics – an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, *219*(3), 671–679.

Delorme, M., & Iori, M. (2019). Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, *109*(3), 1–19.

Delorme, M. (2017). *Mathematical models and decomposition algorithms for cutting and packing problems* [Doctoral dissertation, alma]. http://amsdottorato.unibo.it/7828/

Diegel, A., Miller, G., Montocchio, E., van Schalkwyk, S., & Diegel, O. (2006). Enforcing minimum run length in the cutting stock problem. *European Journal of Operational Research*, *171*(2), 708–721.

Doostmohammadi, M., & Akartunali, K. (2018). Valid inequalities for two-period relaxations of big-bucket lot-sizing problems: Zero setup case. *European Journal of Operational Research*, *267*(1), 86–95.

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research*, *29*(6), 1092–1104.

Dyckhoff, H. (1990). Cutting and packing a typology of cutting and packing problems. *European Journal of Operational Research*, *44*(2), 145–159.

Eiben, A., Michalewicz, Z., Schoenauer, M., & E., S. J. (2007). Parameter control in evolutionary algorithms. *Lobo F.G., Lima C.F., Michalewicz Z. (eds) Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence*, *30*, 19–46.

Eppen, G. D., & Martin, R. K. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, *35*(6), 832–848.

Fiorotto, D. J., Jans, R., & de Araujo, S. A. (2017). An analysis of formulations for the capacitated lot sizing problem with setup crossover. *Computers & Industrial Engineering*, *106*, 338–350.

Foerster, H., & Wäscher, G. (2000). Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research*, *38*(7), 1657–1676.

Gau, T., & Wäscher, G. (1995). Cutgen1: A problem generator for the standard one-dimensional cutting stock problem [Cutting and Packing]. *European Journal of Operational Research*, *84*(3), 572–579.

Gendreau, M., & Potvin, J.-Y. (Eds.). (2010). *Handbook of metaheuristics* (2nd ed.). Springer.

Ghidini, C. T. L. S., Alem, D., & Arenales, M. N. (2007). Solving a combined cutting stock and lot-sizing problem in small furniture industries. *Proceedings of the 6th International Conference on Operational Research for Development (VI-ICORD)*.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, *9*(6), 849–859.

Gilmore, P. C., & Gomory, R. E. (1963). A linear programming approach to the cutting stock problema - part ii. *Operations Research*, *11*(6), 863–888.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69–93.

Golfeto, R. R., Moretti, A. C., & de Salles Neto, L. L. (2009a). A genetic symbiotic algorithm applied to the cutting stock problem with multiple objectives. *Advanced Modeling and Optimization*, *11*(4), 473–501.

Golfeto, R. R., Moretti, A. C., & de Salles Neto, L. L. (2009b). A genetic symbiotic algorithm applied to the one-dimensional cutting stock problem. *Pesquisa Operacional*, *29*, 365–382.

Gonçalves, J. F., & Resende, M. G. C. (2016). Random-key genetic algorithms. *Handbook of heuristics*, 1–13.

Gonçalves, J. F., & Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *J Heuristics*, *17*, 487–525.

Gonçalves, J. F., & Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, *145*(2), 500–510.

Gonçalves, J. F., & Wäscher, G. (2020). A mip model and a biased random-key genetic algorithm based approach for a two-dimensional cutting problem with defects. *European Journal of Operational Research*, *286*(3), 867–882.

Göthe-Lundgren, M., T. Lundgren, J., & A. Persson, J. (2002). An optimization model for refinery production scheduling. *International Journal of Production Economics*, *78*(3), 255–270.

Gramani, M. C. N., & França, P. M. (2006). The combined cutting stock and lot-sizing problem in industrial processes. *European Journal of Operational Research*, *174*(1), 509–521.

Gramani, M. C. N., França, P. M., & Arenales M. (2009). A lagrangian relaxation approach to a coupled lot-sizing and cutting stock problem. *International Journal of Production Economics*, *119*(2), 219–227.

Gruson, M., Bazrafshan, M., Cordeau, J.-F., & Jans, R. (2019). A comparison of formulations for a three-level lot sizing and replenishment problem with a distribution structure. *Computers & Operations Research*, *111*, 297–310.

Haessler, R. W. (1975). Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research*, *23*(3), 483–493.

Haessler, R. W. (1988). Selection and design of heuristic procedures for solving roll trim problems. *Management Science*, *34*, 1460–1471.

Hendry, L. C., Fok, K. K., & Shek, K. W. (1996). A cutting stock and scheduling problem in the copper industry. *The Journal of the Operational Research Society*, *47*(1), 38–47.

Henn, S., & Wäscher, G. (2013). Extensions of cutting problems: Setups. *Pesquisa Operacional*, *33*, 133–162.

Hernandez, W., & Suer, G. (1999). Genetic algorithms in lot sizing decisions. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, *3*, 2280–2286.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

Johnston, R. E., & Sadinlija, E. (2004). A new model for complete solutions to one-dimensional cutting stock problems. *European Journal of Operational Research*, *153*(3), 176–183.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management Science*, *6*(4), 366–422.

Kolen, A. W. J., & Spieksma, F. C. R. (2000). Solving a bi-criterion cutting stock problem with open-ended demand: A case study. *Journal of the Operational Research Society*, *51*(11), 1238–1247.

Krarup, J., & Bilde, O. (1977). Plant location, set covering and economic lot size: An 0 (mn) - algorithm for structured problems. In L. Collatz, G. Meinardus, & W. Wetterling (Eds.), *Numerische methoden bei optimierungsaufgaben band 3* (pp. 155–180, Vol. 36). Birkhäuser Basel.

Luche, J., Morabito, R., & Pureza, V. (2009). Combining process selection and lot sizing models for production scheduling of electrofused grains. *Asia-Pacific Journal of Operational Research*, *26*(3), 421–443.

Ma, N., Liu, Y., & Zhou, Z. (2019). Two heuristics for the capacitated multi-period cutting stock problem with pattern setup cost. *Computers & Operations Research*, *109*(3), 218–229.

Ma, N., Bai, X., & Dong, X. (2021). Hybrid heuristic for the production replanning problem under varying demands in manufacturing industries. *Engineering Optimization*, *0*(0), 1–18.

Ma, N., Liu, Y., Zhou, Z., & Chu, C. (2018). Combined cutting stock and lot-sizing problem with pattern setup. *Computers & Operations Research*, *95*, 44–55.

Martin, M., Yanasse, H. H., & Salles-Neto, L. L. (2022). Pattern-based ilp models for the one-dimensional cutting stock problem with setup cost. *Journal of Combinatorial Optimization*.

Martínez, K. P., Morabito, R., & Toso, E. A. V. (2018). A coupled process configuration, lot-sizing and scheduling model for production planning in the molded pulp industry. *International Journal of Production Economics*, *204*, 227–243.

Martínez, K., Adulyasak, Y., Jans, R., Morabito, R., & Toso, E. (2019). An exact optimization approach for an integrated process configuration, lot-sizing, and scheduling problem. *Computers and Operations Research*, *103*, 310–323.

McDiarmid, C. (1999). Pattern minimisation in cutting stock problems. *Discrete Applied Mathematics*, *98*(1-2), 121–130.

Melega, G. M., de Araujo, S. A., & Jans, R. (2018). Classification and literature review of integrated lot-sizing and cutting stock problems. *European Journal of Operational Research*, *271*(1), 1–19.

Melega, G. M., de Araujo, S. A., & Morabito, R. (2020). Mathematical model and solution approaches for integrated lot-sizing, scheduling and cutting stock problems. *Annals of Operations Research*, *295*(1), 695–736.

Melega, G. M., de Araujo, S. A., Jans, R., & Morabito, R. (2022). Formulations and exact solution approaches for a coupled bin-packing and lot-sizing problem with sequence-dependent setups. *Flexible Services and Manufacturing Journal*.

Mellouli, A., Mellouli, R., & Masmoudi, F. (2019). An innovative genetic algorithm for a multi-objective optimization of two-dimensional cutting-stock problem. *Applied Artificial Intelligence*, *33*(6), 531–547.

Mobasher, A., & Ekici, A. (2013). Solution approaches for the cutting stock problem with setup cost. *Computers Operations Research*, *40*(1), 225–235.

Moretti, A. C., & de Salles Neto, L. L. (2008). Nonlinear cutting stock problem model to minimize the number of different patterns and objects. *Comput. Appl. Math*, *27*, 61–78.

Nemhauser, G., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York: Wiley.

Nonås, S. L., & Thorstenson, A. (2000). A combined cutting-stock and lot-sizing problem. *European Journal of Operational Research*, *120*(2), 327–342.

Nonås, S. L., & Thorstenson, A. (2008). Solving a combined cutting-stock and lot-sizing problem with a column generating procedure. *Computers & Operations Research*, *35*(10), 3371–3392.

Oliveira, W. A., Fiorotto, D. J., Song, X., & Jones, D. F. (2021). An extended goal programming model for the multiobjective integrated lot-sizing and cutting stock problem. *European Journal of Operational Research*, *295*(3), 996–1007.

Pochet, Y., & Wolsey, L. (2006). *Production planning by mixed integer programming*. Springer Science & Business Media.

Poldi, K. C., & de Araujo, S. A. (2016). Mathematical models and a heuristic method for the multiperiod one-dimensional cutting stock problem. *Annals of Operations Research*, *238*(1), 497–520.

Poldi, K. C. & Arenales, M. N. (2010). O problema de corte de estoque unidimensional multiperíodo. *Pesquisa Operacional*, *30*, 153–174.

Poltroniere, S. C., Poldi, K. C., Toledo, F. M. B., & Arenales, M. N. (2008). A coupling cutting stock-lot sizing problem in the paper industry. *Annals of Operations Research*, *157*(1), 91–104.

Quezada, F., Gicquel, C., Kedad-Sidhoum, S., & Vu, D. Q. (2020). A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales. *Computers & Operations Research*, *116*, 104865.

Reinertsen, H., & Vossen, T. W. M. (2010). The one-dimensional cutting stock problem with due dates. *European Journal of Operational Research*, *201*(3), 701–711.

Sadykov, R., & Vanderbeck, F. (2013). Column generation for extended formulations. *EURO Journal on Computational Optimization*, *1*(1), 81–115.

Santos, S. G., de Araujo, S. A., & Rangel, M. S. N. (2011). Integrated cutting machine programming and lot sizing in furniture industry. *Pesquisa Operacional para o Desenvolvimento*, *3*(1), 1–17.

Shi, L., Jiang, Y., Wang, L., & Huang, D. (2014). Refinery production scheduling involving operational transitions of mode switching under predictive control system. *Ind. Eng. Chem. Res.*, *53*, 8155–8170.

Signorini, C. d. A. (2021). *One-dimensional cutting stock problems with multiple period-sapplied to the precast slab industry* [Doctoral dissertation, Universidade Estadual Paulista].

Signorini, C. d. A., de Araujo, S. A., Poltroniere, S., & Melega, G. M. (2022). One-dimensional multi-period cutting stock problem with two stages applied to lattice slab production. *Journal of the Operational Research Society*, -.

Silva, E., Viães, C., Oliveira, J. F., & Carravilla, M. A. (2015). Integrated cutting and production planning: A case study in a home textile manufacturing company. In A. P. F. D. B. Póvoa & J. L. de Miranda (Eds.), *Operations research and big data* (pp. 213–220, Vol. 15). Springer International Publishing.

Silva, E., Melega, G. M., Akartunali, K., & de Araujo, S. A. (2023). Formulations and theoretical analysis of the one-dimensional multi-period cutting stock problem with setup cost. *European Journal of Operational Research*, *304*(2), 443–460.

Song, X., & Bennell, J. A. (2014). Column generation and sequential heuristic procedure for solving an irregular shape cutting stock problem. *Journal of the Operational Research Society*, *65*(7), 1037–1052.

Spears, W. M., & De Jong, K. A. (1991). On the virtues of parametrized uniform crossover. *Proc. 4th Int. Conf. Genetic Algorithms, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufmann*, 230–236.

Suliman, S. M. A. (2012). An algorithm for solving lot sizing and cutting stock problem within aluminum fabrication industry. *International Conference on Industrial Engineering and Operations Management, 2012*.

Tomat, L., & Gradisar, M. (2017). One-dimensional stock cutting: Optimization of usable leftovers in consecutive orders. *Cent Eur J Oper Res*, *25*, 473–489.

Trkman, P., & Gradisar, M. (2007). One-dimensional cutting stock optimization in consecutive time periods. *European Journal of Operational Research*, *179*(2), 291–301.

Umetani, S., Yagiura, M., & Ibaraki, T. (2003). One-dimensional cutting stock problem to minimize the number of different patterns. *European Journal of Operational Research*, *146*(2), 388–402.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, *86*(0), 629–659.

Valério de Carvalho, J. M. (2002). Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, *141*(2), 253–273.

Van Vyve, M., Wolsey, L., & Yaman, H. (2014). Relaxations for two-level multi-item lot-sizing problems. *Math. Program.*, *146*(1-2), 495–523.

Vanderbeck, F. (2000). Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, *48*(6), 915–926.

Vanderbeck, F., & Wolsey, L. A. (2010). *Reformulation and decomposition of integer programs* (M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, & L. A. Wolsey, Eds.). Springer Berlin Heidelberg.

Vanzela, M., Melega, G. M., Rangel, S., & de Araujo, S. A. (2017). The integrated lot sizing and cutting stock problem with saw cycle constraints applied to furniture production. *Computers & Operations Research*, *79*, 148–160.

Villas Boas, B. E., Camargo, V. C. B., & Morabito, R. (2021). Modeling and mip-heuristics for the general lotsizing and scheduling problem with process configuration selection. *Pesquisa Operacional*, *41*(e200000), 1–29.

Wäscher, G., Haussner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*(3), 1109–1130.

Wolsey, L. A. (1977). Valid inequalities, covering problems and discrete dynamic programs. *Annals of Discrete Mathematics*, *1*, 527–538.

Wolsey, L. (1998). *Integer programming*. Wiley. https://books.google.com.br/books?id=x7RvQgAACAAJ

Wu, T., Liang, Z., & Zhang, C. (2018). Analytics branching and selection for the capacitated multi-item lot sizing problem with nonidentical machines. *INFORMS Journal on Computing*, *30*(2), 236–258.

Wuttke, D. A., & Heese, S. H. (2018). Two-dimensional cutting stock problem with sequence dependent setup times. *European Journal of Operational Research*, *265*(1), 303–315.

Yanasse, H. H., & Limeira, M. S. (2006). A hybrid heuristic to reduce the number of different patterns in cutting stock problems. *Computers & Operations Research*, *33*(9), 2744–2756.

Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer London Dordrecht Heidelberg New York.

Zhao, M., & Zhang, M. (2020). Multiechelon lot sizing: New complexities and inequalities. *Operations Research*, *68*(2), 534–551.