UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

HENRIQUE BECKER

The state of the art in MILP formulations for the guillotine 2D knapsack and related problems

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Computer Science

Advisor: Prof. Dr. Luciana Salete Buriol
Coadvisor: Prof. Dr. Olinto Araújo

Porto Alegre
September 2022

CIP — CATALOGING-IN-PUBLICATION

Becker, Henrique

The state of the art in MILP formulations for the guillotine 2D knapsack and related problems / Henrique Becker. – Porto Alegre: PPGC da UFRGS, 2022.

136 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2022. Advisor: Luciana Salete Buriol; Coadvisor: Olinto Araújo.

1. Combinatorial optimization. 2. 2D knapsack. 3. Guillotine cuts. 4. Mathematical formulation. I. Buriol, Luciana Salete. II. Araújo, Olinto. III. Título.

“The optimal solution of a model is not an optimal solution of a problem unless the model is a perfect representation of the problem, which it never is.”

— The Future of Operational Research is Past, Russell L. Ackoff, 1979

DEDICATION AND ACKNOWLEDGMENTS

To my parents for their support and patience, especially in these trying times of forced cohabitation.

To my advisor, Buriol, for all the opportunities given, and for not losing hope.

To my co-advisor, Olinto, for saving my doctorate with a change of topic and one whole year of close assistance.

To Maurício Araldi for the daily dose of gaming and conversation, albeit being on the other side of the atlantic ocean.

To Ana for lending me Maurício this way, our weekend gaming, and shipping discussions.

To my hometown friends for keeping contact (and, therefore, my sanity), and even hopping on board with my online birthday party idea.

To my brother, Artur, for remembering me that the academic journey is tortuous even for the most applied and, also, for every manuscript revised.

To Mateus Martin for lending me his codes of many methods and writing a paper with me. To Dennis and Tadeu which had first shown me what Artur remembered me.

To my college teachers for always seeing in me a future colleague of the profession.

To Marcus Ritt, Manuel Iori, and Reinaldo Morabito, for all their comments related to the proposal of this thesis; special thanks to Marcus for also being the best teacher I knew at UFRGS.

To everyone in the lab, which relied on me for our shared problems and for which I could rely on back; especially Artur, Alberto, and Gabriel, on the matter of our shared experiment servers (both the old servers antics and the new servers purchase).

ABSTRACT

This thesis advances the state of the art in Mixed-Integer Linear Programming (MILP) formulations for Guillotine 2D Cutting Problems by (i) proposing a (re-)formulation that improves on a state-of-the-art formulation by cutting down its size and symmetries; (ii) adapting a previously-known reduction in a novel way for the preprocessing phase of the mentioned formulations; (iii) providing extensive experiments comparing the state of the art and the proposed formulation over many literature datasets; (iv) proposing a hybridised variant of the mentioned formulations which improves the performance for some hard instances; (v) proposing and validating a rotation-only symmetry-breaking strategy for the mentioned formulations. This thesis focuses on the Guillotine 2D Knapsack Problem with orthogonal and unrestricted cuts, constrained demand, unlimited stages, and no rotation. However, the formulation may be adapted to many related problems including the Guillotine 2D Multiple Knapsack Problem, the Guillotine 2D Cutting Stock Problem, and the Guilltone 2D Orthogonal Packing Problem, all three of which are approached and experimented upon in this thesis. The code is available.

Concerning the set of 59 instances used to benchmark the the state-of-the-art formulation in which the author took inspiration, and summing the statistics for all models generated, the proposed formulation has only a small fraction of the variables and constraints of the original model (respectively, 3.07% and 8.35%). The enhanced formulation also takes about 4 hours to solve all instances while the original formulation takes 12 hours to solve 53 of them (the other 6 runs hit a three-hour time limit each). In a recently proposed set of 80 harder instances, the enhanced formulation (with and without the pricing framework) found: 22 optimal solutions for the unrestricted problem (5 already known, 17 new); 22 optimal solutions for the restricted problem (all new for the problem and none is the same as the optimal unrestricted solution); better lower bounds for 25 instances; better upper bounds for 58 instances. Concerning other formulations for the problem in the literature, the proposed formulation has shorter run times, and it proves the optimality for more instances. The proposed formulation only fails to deliver good solutions in the datasets that no formulation was able to solve any instance. In such datasets, other formulations did deliver good primal solutions even if they could not solve any instance.

Keywords: Combinatorial optimization. 2D knapsack. Guillotine cuts. Mathematical formulation.

O estado da arte em formulações de PLI para a mochila guilhotinada 2D e problemas relacionados

RESUMO

Essa tese avança o estado da arte em formulações de Programação Linear Inteira (PLI) para Problemas de Corte Guilhotinado 2D pela (i) propondo uma (re-)formulação que melhora uma formulação do estado da arte por meio da redução do seu tamanho e suas simetrias; (ii) adaptando uma redução já conhecida de forma inovadora para a fase de pré-processamento dessas formulações; (iii) provendo extensivos experimentos comparando o estado da arte e a formulação proposta sobre vários conjuntos de instância da literatura; (iv) propondo uma variante hibridizada das formulações mencionadas que melhora a performance para algumas instâncias difíceis; (v) propondo e validando uma estratégia de quebra de simetrias para as formulações mencionadas. O nosso foco é o Problema da Mochila 2D Guilhotinado com cortes ortogonais e irrestritos, demanda limitada, estágios ilimitados, e sem rotação – entretanto, a formulação pode ser adaptada para vários problemas relacionados incluindo o problema da mochila múltipla 2D guilhotinada, o problema do corte de estoque 2D guilhotinado, e o problema de empacotamento ortogonal 2D guilhotinado, todos os três são abordados e alvo de experimentos nessa tese. O código está disponível.

Considerando as 59 instâncias usadas nos experimentos da formulação em que o autor se inspirou, e somando os valores para todos os modelos gerados, a formulação proposta tem apenas uma pequena fração das variáveis e restrições do modelo original (respectivamente, 3.07% e 8.35%). A formulação melhorada soluciona todas as 59 instâncias em cerca de 4 horas enquanto a formulação original soluciona 53 em 12 horas (as outras 6 instâncias não são solucionadas dentro do limite de 3 horas por instância). Em um conjunto de 80 instâncias difíceis recentemente proposto, a formulação melhorada (com e sem a estrutura de precificação) encontrou: 22 soluções ótimas para o problema com cortes irrestritos (5 já conhecidas, 17 novas); 22 soluções ótimas para o problema com cortes restritos (todas novas para o problema e nenhuma é a mesma que do problema de cortes irrestritos); melhores limitantes inferiores para 25 instâncias; melhores limitantes superiores para 58 instâncias. Considerando outras formulações para o problema na literatura, a formulação proposta apresenta tempos de execução menores, e prova a otimalidade para mais instâncias. Somente nos conjuntos de instâncias em que nenhuma formulação solu-

cionou instância alguma é que a formulação proposta falhou em encontrar boas soluções primais enquanto outras formulações obtiveram êxito. A formulação proposta somente falhou em obter soluções de boa qualidade nos conjuntos de instâncias em que nenhuma formulação conseguiu solucionar instância alguma. Nesses conjuntos de dados, outras formulações obtiveram boas soluções primais mesmo não sendo capazes de solucionar instância alguma.

Palavras-chave: Otimização combinatorial. Mochila 2D. Cortes guilhotinados. Formulação matemática.

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2KP	Two-Dimensional Knapsack Problem
B&B	Branch and bound (noun) and Branch-and-bound (adjective)
G2CSP	Guillotine 2D Cutting Stock Problem
G2KP	Guillotine 2D Knapsack
G2OPP	Guillotine 2D Orthogonal Packing Problem
G2SPP	Guillotine 2D Strip Packing Problem
LB	Lower Bound
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
UB	Upper Bound

CONTENTS

1 INTRODUCTION

The problem this work centres around is the Guillotine 2D Knapsack Problem with orthogonal (and unrestricted) cuts, constrained demand, unlimited stages, and no rotation. The author will refer to this specific variant as G2KP, and fully describe it in the next section. The G2KP is a strongly NP-hard problem (KORF, 2003; DOLATABADI; LODI; MONACI, 2012); more details in Section 7.2. This work also examines a specific kind of restricted cuts, three distinct problems closely related to the G2KP, and the variant that allows piece rotation (in all the studied problems); Other problems and variants may be mentioned to contextualize this work but are not experimented upon. The work focuses on obtaining optimal solutions through Mixed-Integer Linear Programming (MILP).

The three distinct problems mentioned above are the Multiple Knapsack Problem (MKP), the Orthogonal Packing Problem (OPP), and the Cutting Stock Problem (CSP). The Bin Packing Problem (BPP) is also covered, but the author treats it as a special case of the CSP and distinguishes between the two only when their difference (i.e., piece type diversity) becomes relevant. The next section explains the G2KP, its variants, and the basics of the chosen mathematical notation. The other three problems share most of the variants and the notation but are discussed in a separate chapter (Chapter 7).

1.1 Explanation of the G2KP and its variants

An instance of the G2KP consists of: a rectangle of length L and width W (hereafter called *original plate*); a set of rectangles $\bar{J}$ (also referred to as *pieces*) where each rectangle $i \in \bar{J}$ has a length l_i, a width w_i, a profit p_i, and a demand u_i. This work assumes, without loss of generality, that all such values are positive integers.

The G2KP seeks to maximise the profit of the pieces obtained by cutting the original plate. The *guillotine* qualifier means every cut always goes from one side of a plate to the other; a cut never stops or starts from the middle of a plate. The original plate is cut into intermediary plates $j \in J$, $J \supseteq \bar{J}$, which are further cut following the same rule.

If a plate is not cut further, then it may either be: thrown away as trim/waste for no profit; or, if it has the same size as a piece, sold by the piece profit value. *Orthogonal cuts* are always parallel to one side of a plate (and perpendicular to the other). In conjunction with only using guillotine cuts, this means that any intermediary plate j is always

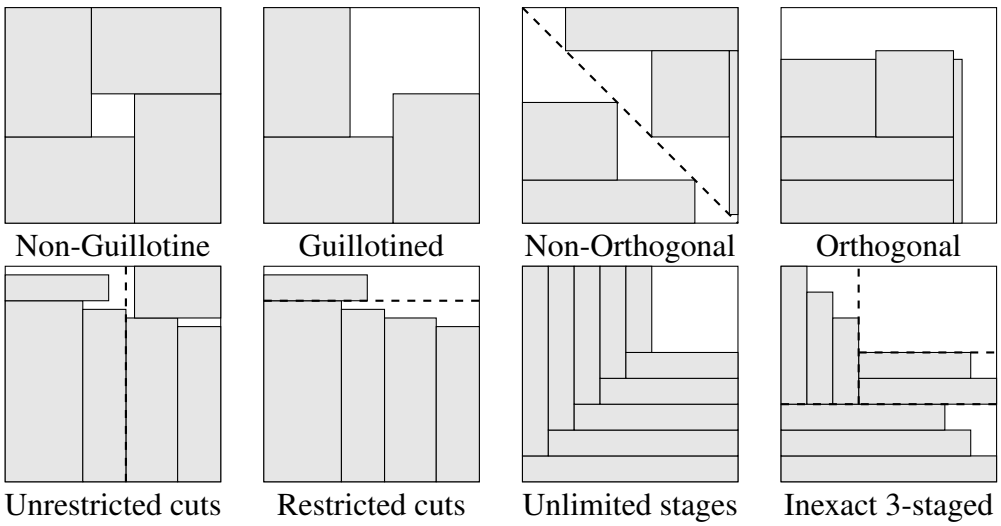
a rectangle, and has a well-defined l_j and w_j. *Unrestricted cuts* mean the machine is allowed to make horizontal (vertical) cuts different from the length (width) of a piece. In contrast, restricted cuts mean horizontal (vertical) cuts can only happen at positions that match a piece length (width), it may also mean that, in addition to this, a piece with matching length (width) *must* be extracted from the first child plate of a restricted cut. In this paper, *restricted* means only that the position of the cuts is restricted (not that the cut force a posterior piece extraction); the author creates and employ the term *position-only restricted* to keep the reader aware of what is meant. Solving the position-only restricted problem exactly is a costly but high-quality primal heuristic for the G2KP.

Constrained demand means at most u_i copies of piece i can be sold, i.e., converted into profit. The G2KP with *unconstrained demand* is not strongly NP-hard, it is weakly NP-Hard; exact algorithms of pseudo-polynomial time complexity exist (BEASLEY, 1985a). Consequently, if $u_i \geq \beta_i : \forall i \in \bar{J}$, where β_i is an upper bound on the number of copies of piece i that can be produced from the original plate, then the instance is probably better solved as an instance of the unconstrained G2KP instead. The author avoids this kind of instance in the experiments. The modifier *unlimited stages* means there is no limit to the number of times the guillotine switches between horizontal and vertical orientations. In the exact k-staged G2KP, the guillotine is switched at most $k-1$ times. Consequently, in a solution of the two-staged G2KP, all cuts in some orientation (and, consequently, parallel to each other) are done before any cuts in the other orientation are done (over the remains of the previous stage). The non-exact k-staged G2KP adds one extra stage in which the only cuts allowed are the ones that trim plates to the size of pieces (i.e., one of the children of this last cut must be waste). The *no-rotation* qualifier means the plates never are rotated as to switch their length and width during the cutting process; especially, a plate j cannot be sold as a piece of length w_j and width l_j.

If the text further qualifies the G2KP, it only means to discard the qualifiers above that directly conflict with the extra qualifiers, if any. For example, suppose the text refers to the *unconstrained G2KP*. In that case, it means only to discard the *constrained* qualifier but keep the remaining qualifiers, i.e., no rotation, unlimited stages, as well as guillotined, orthogonal, and unrestricted cuts. Figure 1.1 may help understand some of the discussed characteristics.

The literature further distinguishes between *weigthed* and *unweighted* problem variants. In the weighted variant, pieces have an arbitrary profit value, while in the unweighted variant, the profit value is always equivalent to the piece area. Consequently,

Figure 1.1 – Examples of valid patterns for most of the discussed problem variants.



In *Non-Orthogonal*, *Unrestricted cuts*, and *Restricted cuts*, the dashed line indicate the first cut of the pattern. The pattern using only restricted cuts packs one less piece because no matter which restricted cut is done first, it is impossible to obtain the same six pieces obtained by the pattern with unrestricted cuts. The initial unrestricted cut is essential to obtain the six-piece pattern. In *Inexact 3-staged*, the dashed lines indicate the last cut of each stage. Consider this last diagram. The bottom three horizontal cuts are done in the first stage. In the second stage, two pieces from the first stage are trimmed by a vertical cut and the three leftmost vertical cuts are done. In the third stage, two of the three pieces of the second stage are trimmed with horizontal cuts and the two rightmost horizontal cuts are done. The fact the variant is inexact only matters for the topmost piece among the two pieces cut in the third stage; the vertical cut that trims that piece *must* be done after the third stage and therefore happens at the trim-only fourth stage only allowed in a three-staged variant if it is *inexact*. Souce: the author.

the unweighted variant is equivalent to minimising waste and is a particular case of the weighted variant. Any algorithm that solves the weighted variant (as is the case in this thesis) can solve the unweighted variant by setting the piece profit values to their areas.

1.2 Motivation

Guillotine cutting problems are of interest of the industry, especially the wood (YANASSE; MORABITO, 2008; MORABITO; BELLUZZO, 2007) and glass cutting industries (CLAUTIAUX et al., 2019; PARREÑO; ALONSO; ALVAREZ-VALDES, 2020), often because of machinery limitations. There is a vast and growing literature on the subject as evidenced by Iori et al. (2020) and by Russo et al. (2020). The cutting optimization problem proposed in the *ROADEF/EURO Challenge 2018* was a guillotine cutting problem. The challenge was developed in collaboration with Saint-Gobain Glass France (a reference on

flat glass manufacture). See Parreño, Alonso and Alvarez-Valdes (2020) for more details on this challenge.

This work focuses on MILP as the solving method (instead of *ad hoc* solutions) because its adaptability amplifies the value of any enhancements discovered. A better MILP formulation means: a better solving procedure for the many (already mentioned) closely related problem variants; a better continuous relaxation for computing an optimistic guess on the objective value of all these variants (some *ad hoc* algorithms of the literature use MILP solvers to compute their bounds); not only a better exact method but also a better base for heuristics or anytime procedures; an immediate benefit from parallelisation, automatic problem decomposition, and solver-implemented heuristics; and, finally, better ageing of the method over the years through the current trends of multiple-cores processors and ever-advancing solver performance.

1.3 Contributions and thesis outline

The contributions of this thesis include:

- an enhanced MILP formulation based on a previous state-of-the-art formulation, its proof of correctness, and empirical evidence of its better performance;
- a novel way to employ a previously known property (plate-size normalisation) for both the original and the enhanced formulations, and empirical evidence of its positive impact on their performance;
- new upper and lower bounds, as well as optimal values, for many recently proposed hard instances from Velasco and Uchoa (2019);
- a direct comparison with recent formulations of the literature highlighting the weak and strong points of each;
- the adaptation of the proposed formulation for three related problems (G2MKP, G2OPP, and G2CSP) and empirical results over literature datasets to serve as base for future comparisons between formulations;
- an hybridisation of the proposed formulation (with the formulation from Silva, Alvelos and Carvalho (2010)) that has moderate success in further reducing the run time for instances in which most time is spent at the B&B phase (by disallowing some symmetries).

For such, the author reimplemented a state-of-the-art MILP formulation and an optional pricing procedure used by it.

The rest of the thesis is organised in the following way. Chapter 2 contextualises the topic of the thesis in the broader literature and lists the prior work on it. Chapter 3 introduces the necessary mathematical concepts as well as the formulation used as the basis for the proposed formulation, the proposed formulation itself, and adaptations for the variant allowing rotation. Chapter 4 experimentally examine the difference in model size and overall performance between the reimplementation of the base formulation, the original data on it, and the proposed (re-)formulation. Chapter 5 experimentally examine the performance of the proposed formulation against other recent formulations of the literature for the G2KP. Chapter 7 explains how to adapt the proposed formulation to three related problems and empirically test the adaptations over literature datasets; it also reports a generation mistake found in the T dataset. Chapter 6 presents an adaptation of the proposed formulation that hybridises it with a prior formulation for a more restricted problem without losing the optimality for the general case; experiments are included. Chapter 8 delivers the conclusions based on the experiments presented in the previous chapters. Appendix A provides a detailed description of each instance dataset employed in this thesis.

2 RELATED WORK

The literature on 2D cutting and packing, in general, is vast, as pointed out by Iori et al. (2020), which catalogues exact methods and relaxations of problems from this field. Even if the scope is further restricted to exact methods for the G2KP, the author could write a forty-page survey on it, as it was done by Russo et al. (2020). Consequently, the author strongly suggests both surveys mentioned above for any reader interested in a broader understanding of the literature.

The review presented here is divided into two parts. The first part contextualises the reader by contraposing aspects of the thesis topic to other aspects found in the broader cutting and packing literature. The second part focus on the brief history of our main topic (i.e., MILP formulations for the G2KP).

2.1 Broader literature contextualisation

This section presents a list of aspects **not** shared by the thesis topic. For each item, the literature concerning that aspect is briefly summarised. If adequate, the item also contrasts the aspect with its opposite present in the main topic.

2.1.1 Non-Guillotined problems

Both the G2KP and the 2KP are strongly NP-hard (IORI et al., 2020), and none is a generalisation of the other. Therefore, theoretically, no problem is overall more challenging than the other. In terms of modelling, however, the 2KP has the additional challenge that, once the first piece is cut, the remaining space is nonconvex (FEKETE; SCHEPERS, 1997). The G2KP has a better subproblem structure in this regard. In the G2KP, once a cut is defined, there are two convex spaces, and the problem may be seen as multiple heterogeneous G2KPs[1]. However, this nicer structure cannot be fully exploited if a 2KP solving method is adapted to support guillotine cuts. In Nascimento, Queiroz and Junqueira (2019), for example, the solving method has a harder time solving the G2KP because it is a method for the 2KP with the overhead of identifying guillotine cuts over it.

[1] In fact, in the Section 7.3.1 it will be shown that adapting the proposed formulation to the Multiple Knapsack Problem (homogeneous variant) is very straightforward because of this property.

2.1.2 Unconstrained problems

The unconstrained G2KP was introduced by Gilmore and Gomory (1965), some mistakes of this first work were posteriorly corrected by Herz (1972)[2] and by Beasley (1985a)[3]. Differently from the constrained variant, which is strongly NP-Hard, the unconstrained variant is weakly NP-Hard (it generalises the 1D Unbounded Knapsack Problem). Consequently, exact solving methods in pseudo-polynomial time are possible, especially through Dynamic Programming (DP). The lack of the combinatorial explosion caused by keeping track of the residual demand is what makes DP the most popular approach for the unconstrained variant and mostly unused for the constrained variant. Such is the gap in difficulty between both variants that the state-of-the-art heuristic for the constrained variant solves the unconstrained variant repeatedly (VELASCO; UCHOA, 2019). The state-of-the-art method for the unconstrained problem appears to be Russo, Sforza and Sterle (2014), which is a DP method; they regarded the B&B algorithm of Kang and Yoon (2011) as the previous state of the art.

2.1.3 Heuristic methods

As it is common for classic (strongly) NP-Hard problems, the literature on heuristic methods is colossal. For example, Ortmann and Vuuren (2010) compares across 252 heuristics for the 2D Strip Packing Problem (guillotine and non-guillotine variants). For the thesis main problem, the G2KP, the author believes the best heuristic method is the one proposed by Velasco and Uchoa (2019). The method is inspired by the dynamic programming state-space relaxation of Christofides and Hadjiconstantinou (1995), which also inspired the earlier state-of-the-art heuristic that is Morabito and Pureza (2010). The procedures described by Velasco and Uchoa (2019) give both lower and upper bounds. These methods were run over 500 instances of the literature for both rotation and no-rotation variants; they obtained either optimal or better bounds in all cases. The bounds

[2]Herz (1972) points out a mistake in an *improved* algorithm presented in Gilmore and Gomory (1965). The improved algorithm reduces computational effort by skipping some horizontal and vertical cuts based on four criteria that should not impact the guarantee of optimality. A counterexample shows that this is not the case as the only optimal solution of the counterexample is unatainable when the four criteria are followed.

[3]Beasley (1985a) points out a mistake in the k-staged recursion presented in Gilmore and Gomory (1965). In their recursion, the cut orientation of the first stage (either horizontal or vertical) changes depending on the current stage, e.g., the third stage assumes the first stage had vertical cuts, but the fourth stage assumes the first stage had horizontal cuts instead.

proved the optimality of 348 instances for the no-rotation variant and 385 for the rotation variant; some instances were open. Velasco and Uchoa (2019) then proposes 80 more challenging instances (which are included in this thesis experiments) and proves the optimality of many of them.

2.1.4 Approximative methods

For the G2KP, with and without rotation or weights, Abed et al. (2015) provide a quasi-Polynomial Time Approximation Scheme (quasi-PTAS), assuming the input data to be quasi-polynomially bounded integers (Adamaszek and Wiese (2014) has a similar result for the non-guillotine variant). For the Guillotine 2D Bin Packing Problem (without rotation), Bansal, Lodi and Sviridenko (2005) gives an Asymptotic PTAS (APTAS). Christensen et al. (2017) informs us that finding a PTAS for the Non-Guillotine 2D Knapsack Problem (with or without rotation or weights) is an open problem and that, for the non-guillotine variant, “Bansal et al. (2009) gave a PTAS for the special case when the range of the profit-to-area ratio of the rectangles is bounded by a constant for both the cases with and without rotations” and “there is no FPTAS unless $P = NP$, even for packing squares into squares (LEUNG et al., 1990). "Gálvez et al. (2017) propose a polynomial-time 1.89-approximation for the Non-Guillotine 2D Knapsack Problem[4], and a polynomial-time $3/2 + \varepsilon$-approximation for the rotation case, improved to a $4/3 + \varepsilon$-approximation if all piece profits are set to one. These are the best results for the respective variants as far as the author knows. For a survey on approximation algorithms for 2D cutting, the author refers to Christensen et al. (2017) and to Iori et al. (2020) (which briefly updates the former).

2.1.5 Restricted cuts

Considering only restricted cuts allows a simplified branching model: instead of having to consider a pseudo-polynomial number of horizontal and vertical cuts over each plate, only $2n$ possibilities need to be considered, two for each piece, the one with the

[4]Gálvez et al. (2017) defines the original plate as a square and the pieces as rectangles. It is possible to employ lossless scaling of the plate and the pieces to, without changing the optimal value, transform the original plate of any problem instance into a square. One downside is an increase in the absolute dimensions (i.e., precision).

horizontal cut first, and the one with the vertical cut first (both generate the respective piece and up to two residual plates). This exact branching model is employed by Silva, Alvelos and Carvalho (2010). For the unconstrained variant, Song et al. (2010), proves a worst-case ratio of at least $6/7$ between restricted and unrestricted optimal solutions for the same instance. In contrast, Furini, Malaguti and Thomopulos (2016), for the constrained variant, puts the ratio at most $5/6$. However, at least for the considered literature datasets, most instances have restricted and unrestricted optimal solutions of the same value. The heuristic for the unconstrained variant described by Song et al. (2010) (which only employs restricted cuts) finds the unrestricted optimal value in "94.84%, 86.67% and 77.83% for small, medium and large sized unweighted instances" and "99.67%, 99.50% and 97.00% for small, medium and large sized weighted instances". Furini, Malaguti and Thomopulos (2016) found that 47 of 50 instances, solved optimally by both restricted and unrestricted MILP models, shared the optimal solution value.

2.1.6 Limited number of guillotine stages

Unrestricted cuts are unnecessary to obtain any optimal solution of a two-staged variant (exact or non-exact), so the distinction between restricted and unrestricted two-staged variants do not exist. For k-staged variants in which $k \geq 3$ there is such distinction, e.g., Puchinger and Raidl (2007) first models a formulation for the restricted three-staged G2CSP to then extend it to the unrestricted three-staged G2CSP. For the unconstrained variant, and a small dataset, Beasley (1985a) reported an average of about 0.4% difference between the two-staged optimal value and the unlimited stages optimal value (about the same between two and three-staged, because three-staged is only 0.02% behind unlimited stages). For the constrained variant, however, Martin et al. (2020a) presents an average difference of 3.6% between two-staged and unlimited stages optimal solution values. None of these papers poses this difference as a research question; these percentages are the byproduct of data gathered to answer other questions. For the Guillotine 2D Bin Packing Problem, there is a theoretical result stating a two-staged optimal solution value may be at most 1.691 times worse than a solution with unlimited stages, but this is not a tight bound (BANSAL; LODI; SVIRIDENKO, 2005).

2.1.7 Exact but not pure MILP

Besides solving methods with no relation to MILP models, this section also includes techniques that use MILP models but interleave their use with indispensable calls to non-MILP methods. This section concentrates on works tackling the G2KP, and it is noted if another problem is considered instead. The exact procedures observed here often fall into three categories: (i) branch-and-bound (or, as reported in the seminal works, *tree search*); (ii) graph-based algorithms; (iii) repeated piece subset selections and packing tentatives. The first category (i.e., B&B) may be divided into top-down and bottom-up approaches. Top-down approaches start from the original plate and branch on cuts over it and its subdivisions; this approach is probably the oldest one and is used in Christofides and Whitlock (1977) and in Christofides and Hadjiconstantinou (1995). Bottom-up approaches start from the pieces and combine them into builds, and the builds with each other, while they are smaller than the original plate. This approach has many examples: Viswanathan and Bagchi (1993), Hifi (1997), Cung, Hifi and Cun (2000), and Yoon, Ahn and Kang (2013). The graph-based approaches include Morabito and Arenales (1996) and Clautiaux et al. (2018) (designed for four-staged but tested on unlimited stages, too); many previous papers by Clautiaux use graph representations for the G2OPP.

Finally, there are methods (often aided by MILP solvers) that solve a problem to select the most profitable subset of pieces, and then check if the subset can be guillotine-packed. Dolatabadi, Lodi and Monaci (2012) uses this approach, as did Pisinger and Sigurd (2007), but the latter focus on the G2CSP and the G2KP is solved just as a subproblem. Russo et al. (2020) consider the methods of both Dolatabadi, Lodi and Monaci (2012) and Yoon, Ahn and Kang (2013) to be state of the art, but the same work also points out that both methods have bounding flaws that may lead to incorrect results.

Comprehensive counterexamples for the three bounding flaws found are given in the appendix of Russo et al. (2020). A brief overview of the roots of these three flaws follows. The *first two flaws* one in the antiredundancy strategy D1 of Cung, Hifi and Cun (2000) and another in the unnamed strategy from Dolatabadi, Lodi and Monaci (2012) are, in essence, manifestations of the same oversight. The oversight consists of restricting the search for an optimal solution to combinations of optimal solutions of subproblems (which are the same as the original problem but with the original plate dimensions reduced). Such strategy is the basis of all dynamic programming and works flawlessly for the *unconstrained* G2KP. However, in the constrained case, a suboptimal solution to a

subproblem may be necessary to assemble the optimal solution to the original problem. A suboptimal solution for a subproblem may be necessary because, if an optimal solution for the subproblem were used instead, the combined solution would end up with more copies of some piece type than allowed by the piece demand, immediately becoming an invalid solution. Removing pieces of the combined solution until the number of copies respects the demand does not (always) solve the problem; it is possible that the combination of two suboptimal solutions for the same subproblems would have a better value than this partially dismantled combination of optimal solutions. The *third flaw* was found in the works of G, Seong and Kang (2003), Kang and Yoon (2011), Yoon, Ahn and Kang (2013) and consists on a rounding problem. Two formulae expressing upper bounds on the optimal solution value (i.e., optimal profit) have each two terms rounded down before they are summed. Russo et al. (2020) presents a counterexample in which both *upper* bounds give a result one unit *lower* than the optimal solution value. The bounds are valid if the rounding down only happens after the sum and not before it. The method of Yoon, Ahn and Kang (2013) also carries the flaw from Cung, Hifi and Cun (2000), which has affected other bottom-up methods in the literature too (see Russo et al. (2020) for a complete overview). Consequently, there is no clear definition of which is the best method currently.

2.2 MILP formulations for the G2KP

Russo et al. (2020) identify three strategies employed by previous exact methods which cause loss of the optimality guarantee, i.e., these methods cannot be considered exact anymore. No previous MILP formulation, nor this work, employs any of these strategies. One of these strategies is a dominance rule valid for the unconstrained case but not for the constrained one. Interestingly, Herz (1972) proposed a dominance rule for the unconstrained G2KP based on the same principle and warned about the possibility of misusing the rule in the constrained case.

The first MILP formulation dealing with guillotine cuts and unlimited stages was proposed by Messaoud, Chu and Espinouse (2008). The problem considered by Messaoud, Chu and Espinouse (2008) is the Strip Packing Problem[5], but adapting the for-

[5]The Strip Packing Problem is a two-dimensional cutting/packing problem in which the pieces do not have profit values, and the original plate does not have a predefined length ('height' in the context of the problem); the objective is to minimize the height of the original plate while packing every piece.

mulation to the knapsack variant would not change its fundamentals. Previously, Lodi and Monaci (2003) had proposed two MILP formulations for two-staged G2KP. As noted by Belov (2003), modeling k-staged cuts for $k \geq 3$ (unlimited stages included) was considered difficult at the time. The size of most k-staged formulations is exponential on the number of stages (i.e., k). The formulation of Messaoud, Chu and Espinouse (2008) had about $3n^4/4$ variables and $2n^4$ constraints (where n is the number of pieces); it also employed, according to the authors, a "very loose linear relaxation" due to which "the practical interest of this formulation is still limited". The characterization of guillotine cuts proposed by Messaoud, Chu and Espinouse (2008) seems to have been simultaneously proposed by Pisinger and Sigurd (2007).

The first MILP formulation specifically for the G2KP was proposed by Furini, Malaguti and Thomopulos (2016). The formulation is classified as an extension of the one-cut formulation from Dyckhoff (1981) for the one-dimensional Cutting Stock Problem. However, the formulation from Silva, Alvelos and Carvalho (2010) can be seen as an intermediary step between these two: it had already extended the one-cut formulation for two dimensions in a similar fashion but did not alter the problem from the cutting stock and was limited to two stages and restricted three stages. An extended version of Furini, Malaguti and Thomopulos (2016) appears in Thomopulos (2016) (a PhD thesis), and a prelude to it in Furini and Malaguti (2016). Their formulation has pseudo-polynomial size, $O((L+W) \times L \times W)$ variables and $O(L \times W)$ constraints, and its relaxation provides a stronger bound than Messaoud, Chu and Espinouse (2008). It was the first formulation able to solve medium-sized instances of the literature. Besides the formulation, Furini, Malaguti and Thomopulos (2016) proposes two reductions and one pricing procedure; these three are reimplemented in this work. They also present and prove a theorem to assure the correctness of one of their reductions (*Cut-Position*). A similar theorem and proof appear in Song et al. (2010) but for the unconstrained variant.

In this work, the author proposes an enhanced formulation based on the formulation from Furini, Malaguti and Thomopulos (2016) mentioned above. A significant advantage of the enhancement is to avoid the enumeration of any cuts after the middle of a plate. This advantage appears in many works since Herz (1972). Recently, Delorme and Iori (2019) enhanced a pseudo-polynomial formulation for the 1D Cutting Stock Problem to obtain this same advantage. However, the way Delorme and Iori (2019) changes their formulation to obtain this advantage is different from the approach taken here. The mechanisms involved in each approach are distinct. Delorme and Iori (2019) mechanism

adds only a single reflecting dummy node per bin stock size to compensate for the removed arcs/variables, but it is meant for the 1D variant of the problem. The mechanism proposed here adds a set of extraction variables in which the cardinality depends on the geometric properties of the specific instance considered, but it is meant for the 2D variant and also cuts down some symmetries that only exist in the 2D variant (i.e., are not present in the 1D variant).

The most recent MILP formulations for the G2KP come from the following three works by Martin et alii: Martin et al. (2020a), Martin, Morabito and Munari (2020a), Martin, Morabito and Munari (2020b). Besides the formulations from these three works, a previous formulation from Martin et al. (2019) targets the G2KP with defects (i.e., the original plate has regions that cannot be part of any piece). A direct comparison between formulations with and without defects is unfair; modelling the defects burdens a formulation greatly. Moreover, most formulations cannot be straightforwardly adapted to the G2KP with defects, including the one proposed in this work. However, the results may be interpreted not as a direct comparison, but as an assessment of the impact of modelling defects over a formulation. These four formulations are discussed in details in Chapter 5 in which they participate in experiments.

3 TECHNICAL BACKGROUND AND THE PROPOSED FORMULATION

This chapter introduces all the technical details necessary to contextualize the proposed formulation and the formulation itself. Outside of this chapter, the proposed formulation (also referred to as the enhanced formulation) will also be referred to as BBA (Becker, Buriol, and Araújo) as to follow the same naming convention applied to FMT (Furini, Malaguti, and Thomopulos) from Furini, Malaguti and Thomopulos (2016). Section 3.1 introduces notation for the chosen discretisation, discusses some alternative discretisations, outlines the cut-and-plate enumeration procedure, and dives deeply into the known (but innovatively employed) plate-size normalisation. Section 3.2 summarises the formulation proposed in Furini, Malaguti and Thomopulos (2016), which is the basis for the formulation proposed by this thesis and gives it an intuitive interpretation. Section 3.3 describes the proposed formulation, how it differs from its basis, and intuition for why the changes (generally) lead to a performance improvement. Section 3.4 presents a proof for the claim that the proposed formulation can find a cutting pattern for any multiset of pieces for which the original formulation also finds a cutting pattern, this is, that the guarantee of optimality is retained. Section 3.5 shows how both discussed formulations can be adapted to allow piece rotation and Section 3.6 shows a reduction that is specific to those formulations with rotation enabled. Finally, Section 3.7 presents a summary of the pricing procedure employed in Furini, Malaguti and Thomopulos (2016) and reimplemented by the author for the experiments of this thesis.

3.1 Notation, Discretisation, and Plate-Size Normalisation

The performance of solving methods for cutting and packing problems often heavily depends on the number of cut/packing positions considered. Since the seminal works of Christofides and Whitlock (1977) and Herz (1972), solving methods avoid considering each possible position but instead consider only a subset necessary to guarantee optimality. The literature includes many such subsets, which are often referred to as *discretisations*. The most common way of computing these discretisations is Dynamic Programming (DP) algorithms. These DP algorithms usually only take a small fraction of the running time, but the size of the position subset outputted by them strongly affects the time spent by the rest of the solving method.

Both the proposed formulation and its basis have one constraint for each attainable

distinctly-sized plate and one variable for each potential cut over each of these plates. Therefore, eliminating a single cutting position has the following effects: **(i)** it removes one variable for each distinctly-sized plate that allowed that cutting position; **(ii)** if that cutting position was the only way to produce some distinctly-sized plates[1], then it also removes the constraints associated with these plates; **(iii)** if (ii) excludes one or more constraints/plates, then it also excludes all variables representing possible cuts over the excluded plates; **(iv)** finally, if (iii) eliminates one or more variables/cuts, then it may trigger (ii) again (i.e., other plates stop being attainable), cyclically.

In this work, the only cut subset (discretisation) considered are the canonical dissections of Herz (1972), hereafter referred to as *normal cuts* instead. The author acknowledges the existence of stricter discretizations: the raster points of Terno, Lindemann and Scheithauer (1987), Scheithauer and Terno (1996), the regular normal patterns of Boschetti, Mingozzi and Hadjiconstantinou (2002) (named this way by Côté and Iori (2018)), and the Meet-in-the-Middle (MiM) of Côté and Iori (2018). The reasons for the author's choice of discretisation are numerous: it works well with the *Plate-Size Normalisation* procedure described in the sequence; it is the same discretisation employed by Furini's formulation (from which the proposed formulation is based on);The main gain of MiM is reducing the number of cut positions after the middle of a plate, which the enhanced formulation already discards anyway; the regular normal patterns compute a distinct subset-sum for each pair of plate and piece, which the author considers excessive (there may exist hundreds of thousands of intermediary plate possibilities); finally, the raster points complicate the proofs presented here and the *Plate-Size Normalisation* weakens its benefits.

The set $O = \{h, v\}$ denotes the cut orientation: h is horizontal (parallel to width, perpendicular to length); v is vertical (parallel to length, perpedicular to width). Let us recall that the demand of a piece $i \in \bar{J}$ is denoted by u_i. By defining the set of pieces fitting a plate j as $I_j = \{i \in \bar{J} : l_i \leq l_j \wedge w_i \leq w_j\}$, the set N_{jo} (i.e., the set of the normal cuts of orientation o over plate j) can be defined as:

$$N_{jo} = \begin{cases} \{q : 0 < q < l_j; \quad \exists n_i \in [0 \,..\, u_i], \forall i \in I_j, q = \sum_{i \in I_j} n_i l_i\} & \text{if } o = h, \\ \{q : 0 < q < w_j; \quad \exists n_i \in [0 \,..\, u_i], \forall i \in I_j, q = \sum_{i \in I_j} n_i w_i\} & \text{if } o = v. \end{cases} \tag{3.1}$$

[1]Note that the same cutting position, when applied to distinctly-sized plates, may generate different children.

The sets defined above never include cuts at the plate extremities (i.e., 0, l_j for N_{jh}, and w_j for N_{jv}). Any of these cuts will always create (i) a zero-area plate and (ii) a copy of the plate that is being cut. Consequently, these cuts only add symmetries and may be disregarded.

The set J can now be defined by the following procedure: the original plate (plate 0) is added to J, then for every plate $j \in J$ every cut in $N_{jv} \cup N_{jh}$ is applied to j, and each child generated is added to J if it can fit at least one piece. The process finishes when every plate in J was considered for cutting, and no new plates were generated. Such procedure guarantees each piece $i \in \bar{J}$ will always be present in J unless the piece does not fit the original plate (in which case it is irrelevant to the problem and could be removed a priori).

The goal of the *Plate-Size Normalisation* procedure propose here is to reduce the number of distinctly-sized plates considered. Fewer distinctly-sized plates mean fewer constraints and trigger the same cascading effect described by items (ii)–(iv) above. The property exploited by the procedure is already known and exploited by Alvarez-Valdes, Parreño and Tamarit (2009) and by Dolatabadi, Lodi and Monaci (2012). The author chose to state the property the following way:

Proposition 3.1 *Given a plate $j \in J$, l_j may always be replaced by $l'_j = max\{q : q \in N_{kh}, q \leq l_j\}$ in which $k \in J$, $w_k = w_j$, but $l_k > l_j$, without loss of optimality. The analogue is valid for the width.*

In other words, if increasing the length (width) of plate j reveals that the original length (width) did not match a normal cut position in the enlarged plate, then plate j may be replaced by a shorter plate in which the length (width) is reduced to the largest normal cut position smaller than the original length (width). For example, given $l = [5, 7]$, $w = [3, 2]$, a 13x3 plate may be reduced to 12x3 (13 does not match a normal cut while $5 + 7 = 12$ does), and a 13x2 plate may be reduced to 7x2 (13 does not match a normal cut while 7 does). No proof is replicated here. The following can now be defined:

Definition 3.1 (Size-normalised plate) *The length of a plate j is considered normalised if, and only if, $l_j = l'_j$. The analogue is valid for the width. The size of a plate is normalised if, and only if, both its length and its width are normalised.*

The *Plate-Size Normalisation* procedure proposed here consists only of replacing every non-size-normalised plate enumerated by its normalised counterpart. In summary,

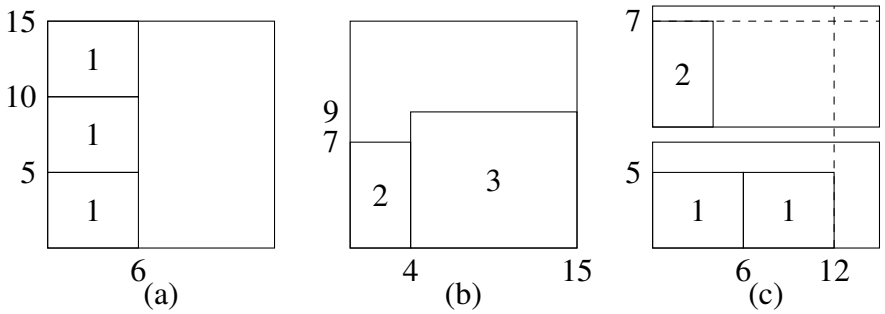
every intermediary plate *for which the length (width) is not a linear combination of the length (width) of the pieces* have its dimensions reduced to the closest linear combination smaller than itself; this combines multiple plate types that could only pack the same set of pieces into a single plate type. The only extra effort added by *Plate-Size Normalisation* consists of binary searches over N_{jo} sets for each plate j, and these may be carried out without increasing the overall complexity, given the setup of $O(LW)$ vectors of size $O(L+W)$; a setup step which also does not increase the overall complexity. However, in the implementation, the author opted to increase the overall complexity from $O(L^2W + LW^2)$ to $O(L^2Wlog(L) + LW^2log(W))$ because the fraction of time spent on the enumeration was not enough to justify the memory and code complexity trade-off. In practice, even if the worst-case complexity increases, the time spent decreases because the actual number of plates (denoted in the complexity by $O(LW)$ becomes more distant from the worst-case. A suitable N_{ko} set for each plate j was already computed by the plate enumeration procedure before introducing the *Plate-Size Normalisation* (no extra effort required).

Remark 3.1 *If a normal cut divides a size-normalised plate, then the dimension perpendicular to the cut, in the first child, is normalised. The dimension parallel to the cut in the first child, and both dimensions of the second child, are not guaranteed to be normalised.*

Example 3.1 (Denormalisation after normalised cut) *Consider three pieces with* $l = [5, 7, 9]$, $w = [6, 4, 11]$, $u = [3, 1, 1]$, *and a plate of dimensions* 15x15. *The plate dimensions are already normalised. The plate length matches stacking the three copies of the first piece. The plate width matches the other two pieces lying side-by-side. An horizontal cut at length* 7 *is a normal cut because it matches the length of the second piece. If the cut is done, the width of both children is not normalised anymore, nor is the length of the second child. The width of both children is not normalised because the third piece does not fit either child so, for both children, the largest width a valid packing may reach is* 12. *The length of the second child is not normalised because the largest length a valid packing inside the second child may reach is 7. The dimensions of both children may be normalised to* 7x12. *This example already shows an immediate gain: instead of creating two new plate sizes, the enumeration only creates a single new plate type. The cut creates two copies of this single type of plate.*

While the basic property (proposition 3.1) is well known by the literature, some of the developments given here are not. Generally, in the literature, the procedure is em-

Figure 3.1 – Diagram of Example 3.1



Notes about the diagrams: (a) the three copies of the first piece stacked; (b) the second and third pieces side-by-side; (c) both children of an horizontal normal cut over a normalised plate are not normalised themselves.

ployed a single time just for the original plate, and *not* for every plate generated during an enumeration procedure like it is done here. Because the procedure is commonly applied once, the denormalisation effect mentioned here is not studied or discussed. Also, applying the procedure a single time often is done to get a better relaxation, which is not the case for the proposed formulation. In the proposed formulation and its basis, the dimensions are abstracted by a flow graph connecting cuts and plates; the dimensions are not right-hand values of binding constraints. The plate-size normalisation does not affect the relaxation; instead, it reduces the model size because multiple intermediary plate types are conflated into a single type.

3.2 The FMT formulation and associated reductions

Given pseudo-polynomial time and space, an instance of the G2KP can be transformed into a bipartite directed acyclic (multi)graph; solving a flow-like problem over such graph is equivalent to solving the original G2KP instance. The two disjoint and independent sets of vertices are (i) the enumerated plate types and (ii) the enumerated cuts over the plate types. Each cut vertice has one incoming edge and one or two outgoing edges. The head of the incoming edge is the plate vertice that represents the plate being cut. The tail of each outgoing edge is a plate vertice representing a plate produced by the cut. These are all edges that exist in the graph. If the cut vertice has only two incident edges, it represents a trim cut, i.e., a cut that only reduces the size of an existing plate without producing a second plate. If the cut vertice has three incident edges, it represents a plate cut into two smaller plates. As the graph is a multigraph, it allows for parallel

edges, representing a cut exactly at the middle of a plate generating two copies of the same plate type.

The aforementioned flow-like problem is as follows. All edges only allow integer amounts to flow between vertices. The vertice representing the original plate type is the only one to start with one flow unit (all other vertices start zeroed). If a plate vertice receives any flow amount, it can keep any portion of the flow in the vertice and freely redistribute the remaining flow among its outgoing edges. If a cut vertice receives any flow amount, it *multiplies* the amount of flow received by the number of outgoing edges, and *must* relay the exact amount of flow received to *each* of the outgoing edges, e.g., if a cut vertice receives two units of flow then each outgoing edge receives two units of flow. If a plate vertice represents a plate type of the same dimensions as a piece i, then each unit of flow kept by the vertice generates a profit p_i constrained to a maximum of $u_i \times p_i$. The problem is deciding how the plate vertices will distribute the flow they receive to maximize said profit.

The formulation proposed in Furini, Malaguti and Thomopulos (2016), henceforth referred to as the FMT formulation, generates models similar to the graph described and which are solved similarly to the flow-like problem mentioned. As previously defined, the set $O = \{h, v\}$ denotes the horizontal and vertical cut orientations. The set Q_{jo} ($\forall j \in J, o \in O$) denotes the set of possible cuts (or cut positions) of orientation o over plate j. The set Q_{jo} is not formally defined because it is a subset of N_{jo} (formally defined in the last section) that varies based on which reductions are being applied.

The parameter a is a byproduct of the plate enumeration process and represents the edges of the graph. The value of a^o_{qkj} indicates how many copies of a plate $j \in J$ are produced by cutting a plate $k \in J$ with a cut of orientation $o \in O$ at position $q \in Q_{ko}$. This value may be zero (no plate created by the cut has the same dimensions as j, i.e., no edge exists), one (one plate created by the cut has the same dimensions as j, i.e., there is an edge), or two (the parent plate was cut in half, and both halves have the same dimensions as j, i.e., two parallel edges). This parameter is needed to write the constraints that control which plates are available. The description of this parameter in Furini, Malaguti and Thomopulos (2016) has a typo, as pointed out by Martin et al. (2020b):"[...] there is a typo in their definition of parameter a^o_{qkj}, as the indices j and k seem to be exchanged.". The original parameter description also forgets the possibility that it may have value two (instead of just zero and one).

In a valid solution, the value of x^o_{qj} is the number of times a plate $j \in J$ is cut with

orientation $o \in O$ at position $q \in Q_{jo}$; i.e., how much flow is being transported by each edge coming from a plate vertice. The plate $0 \in J$ is the original plate, and it may also be in $\bar{J}$, as there may exist a piece of the same size as the original plate. The y_i variable denotes the number of times a plate i was sold as the piece i (as $\bar{J} \subseteq J$, each index $i \in \bar{J}$ denote both a piece and the plate of the exact same dimensions).

$$\boldsymbol{max.} \sum_{i \in \bar{J}} p_i y_i \tag{3.2}$$

$$\boldsymbol{s.t.} \sum_{o \in O} \sum_{q \in Q_{jo}} x^o_{qj} \leq \sum_{k \in J} \sum_{o \in O} \sum_{q \in Q_{ko}} a^o_{qkj} x^o_{qk} \qquad \forall j \in J \setminus \bar{J}, j \neq 0, \tag{3.3}$$

$$y_i + \sum_{o \in O} \sum_{q \in Q_{io}} x^o_{qi} \leq \sum_{k \in J} \sum_{o \in O} \sum_{q \in Q_{ko}} a^o_{qki} x^o_{qk} \qquad \forall i \in \bar{J}, i \neq 0, \tag{3.4}$$

$$\sum_{i \in \{i | i \in \bar{J} \wedge w_i = w_0 \wedge l_i = l_0\}} y_i + \sum_{o \in O} \sum_{q \in Q_{0o}} x^o_{q0} \leq 1 \qquad , \tag{3.5}$$

$$y_i \leq u_i \qquad \forall i \in \bar{J}, \tag{3.6}$$

$$x^o_{qj} \in \mathbb{N}^0 \qquad \forall j \in J, o \in O, q \in Q_{jo}, \tag{3.7}$$

$$y_i \in \mathbb{N}^0 \qquad \forall i \in \bar{J}. \tag{3.8}$$

The objective function maximizes the profit of the plates sold as pieces (3.2). Constraints (3.3) and (3.4) guarantee that for every intermediary plate j that was further cut (left-hand side), or plate/piece i that was either sold or further cut (left-hand side), there must be a cut making available a copy of such plate (right-hand sides). One copy of the original plate is available from the start (3.5), and it can be either sold as a piece of the same dimensions or further cut. The amount of sold copies of some piece type must respect the demand for that piece type (3.6). Finally, the domain of all variables is the non-negative integers (3.7)-(3.8).

Besides the base formulation, Furini, Malaguti and Thomopulos (2016) also employs a basic symmetry-breaking strategy and proposes two reductions for the model size. The cut enumeration in Furini, Malaguti and Thomopulos (2016) excludes one of each pair of perfectly symmetrical cuts. Perfectly symmetrical cuts are pairs of cuts that create the same set of two child plates (i.e., just which is the first and which is the second that is reversed). Furini, Malaguti and Thomopulos (2016) chose to ignore the symmetric cut in the second half of the plate during the enumeration. The *Cut-Position* reduction confines the set of cutting positions over some plates to the restricted set. The condition for a plate

to be affected by Cut-Position is that it should be able to pack *at most* five pieces because, in such case, its restricted and unrestricted optimal value are the same (see Figure 6.2). The *Redundant-Cut* reduction only removes a subset of the *trim cuts*, i.e., cuts in which the second child plate is immediately considered waste because it is smaller than every piece. Redundant-Cut removes a trim cut over plate j, which obtains plate j', if plate j itself can *only* be obtained by trim cuts of the same orientation from larger plates. The rationale is that these larger plates can directly obtain j' through a single trim cut instead of using j as an intermediary plate. This way, the reduction removes unnecessary intermediaries and avoids the many symmetric patterns that only vary on the number of trim cuts employed to reduce a larger plate down to j'.

3.3 The proposed (re-)formulation

The FMT is elegant: the pieces are just intermediary plates that may be sold. The proposed changes affect both the enumeration step and the formulation mathematical description. These changes significantly reduce the model size. However, these changes also deepen the distinction between plates and pieces and may be regarded as sacrificing some elegance for performance. The essentials of the formulation remain the same, but the proposed formulation has better performance in the vast majority of the problem instances. For this reason, the author considers the formulation proposed here as an enhanced iteration of the FMT. In general, the relaxation of the proposed model is the same as the FMT.

As mentioned at the end of the last section, FMT allows for removing one of each pair of perfectly symmetrical cuts (which can be arbitrarily chosen to be the one in the second half of the plate). Differently, Christofides and Whitlock (1977) disregards *all* cuts after the middle of the plate because of symmetry. If FMT did the same as Christofides and Whitlock (1977), it could become impossible to trim a plate to the size of a piece. For example, if there was a piece with a length larger than half the length of a plate, and such plate has no normal cut with the exact length of the needed trim, then the piece could not be extracted from the plate, even if the piece fits into the plate. The goal of the proposed formulation is to reduce the number of cuts (i.e., model variables) by getting closer to the symmetry-breaking rule used in Christofides and Whitlock (1977) without loss of optimality. Of course, disregarding every cut after the midplate also insulates against any perfectly symmetrical cuts and, therefore, supersedes a check just for this specific kind of

symmetry.

Considering the graph representation for FMT presented in the last section, it can be said that the proposed formulation throws away the possibility of piece-sized plate vertices to convert flow into profit and creates a third disjoint and independent set of vertices representing the pieces. The vertices of this new set are all leaves/sinks responsible for converting flow to the corresponding piece profit. The edges that reach the new vertices always come from plate vertices, but the plates do not need to have the exact dimensions of the pieces anymore. These edges work as a shortcut to trimming, and multiple vertice plates may have an edge pointing to the same piece vertice as well as each vertice plate may have edges to multiple piece vertices. This change alone leads to a larger model; however, as the edges to this new set of vertices allow us to obtain pieces without needing trim cuts, the symmetry-breaking strategy can be expanded to consider every cut after the middle of a plate. This reduction leads to far fewer edges, as the cut position discretisation is often denser in the second half of the plate (compared to the first half). It also means fewer plate vertices, as many plates were only necessary as intermediary steps to trimming a plate to the size of a piece.

3.3.1 Changes to the formulation description

The changes to the formulation are restricted to replacing the set of integer variables $y_i, i \in \bar{J}$, with a new set of variables $e_{ij}, (i,j) \in E, E \subseteq \bar{J} \times J$, and the necessary adaptations to accomodate this change. In FMT, y_i denotes the number of times a plate i was sold as the piece i (plate i has the exact same dimensions as piece i). The *extraction variables* e_{ij} denote a piece i was extracted from plate j which dimensions may only be the same or larger than the piece i dimensions. The exact definition of set E is discussed over Section 3.3.2; for the purpose of presenting the formulation, the intuitive definition of e_{ij} just above is enough. For convenience, the following are defined: $E_{i*} = \{j : \exists\, (i,j) \in E\}$ and $E_{*j} = \{i : \exists\, (i,j) \in E\}$. The variables x^o_{qj} and coefficients a^o_{qkj} have the same meaning as the FMT formulation (Section 3.2).

$$\boldsymbol{max.} \sum_{(i,j)\in E} p_i e_{ij} \tag{3.9}$$

$$\boldsymbol{s.t.} \sum_{o\in O}\sum_{q\in Q_{jo}} x^o_{qj} + \sum_{i\in E_{*j}} e_{ij} \leq \sum_{k\in J}\sum_{o\in O}\sum_{q\in Q_{ko}} a^o_{qkj} x^o_{qk} \qquad \forall j \in J, j \neq 0, \tag{3.10}$$

$$\sum_{o\in O}\sum_{q\in Q_{0o}} x^o_{q0} + \sum_{i\in E_{*0}} e_{i0} \leq 1 \qquad , \tag{3.11}$$

$$\sum_{j\in E_{i*}} e_{ij} \leq u_i \qquad \forall i \in \bar{J}, \tag{3.12}$$

$$x^o_{qj} \in \mathbb{N}^0 \qquad \forall j \in J, o \in O, q \in Q_{jo}, \tag{3.13}$$

$$e_{ij} \in \mathbb{N}^0 \qquad \forall (i,j) \in E. \tag{3.14}$$

The objective function maximizes the profit of the extracted pieces (3.9). Constraint (3.10) guarantees that for every plate j that was further cut or had a piece extracted from it (left-hand side), there must be a cut making available a copy of such plate (right-hand side). One copy of the original plate is available from the start (3.5), and it can be either have a piece directly extracted or be further cut. The amount of extracted copies of some piece type must respect the demand for that piece type (a piece extracted is a piece sold) (3.12). Finally, the domain of all variables is the non-negative integers (3.13)-(3.14).

3.3.2 Changes to the cut-and-plate enumeration

As mentioned in Section 3.2, the FMT was proposed together with some mechanisms to reduce the model size: removing perfectly symmetrical cuts, *Cut-Position*, and *Redundant-Cut*. Cut-Position does not conflict with the proposed formulation and can be employed alongside it to further reduce the model size. The symmetry-breaking and Redundant-Cut are superseded in the proposed formulation. Redundant-Cut eliminates a subset of the trim cuts from the formulation; however, the proposed formulation does not have trim cuts like those removed by Redundant-Cut because the extraction variables make them unnecessary.

The use of the x variables does not change from the original formulation to the revised formulation – however, the size of the enumerated set of variables changes. The revised enumeration does not create any variable x^o_{jq} in which $(o = h \ \wedge \ q > \lceil w_j/2 \rceil) \vee (o = v \ \wedge \ q > \lceil l_j/2 \rceil)$.

The original formulation has variables y_i, $i \in \bar{J}$, while the revised formulation replaces them with variables e_{ij}, $(i,j) \in E$, $E \subseteq \bar{J} \times J$. Set $\bar{J} \times J$ is orders of magnitude larger than $\bar{J}$. Consequently, set E must be a small subset to avoid having a revised model with more variables than the original. A suitable subset may be obtained by a simple rule: $(i,j) \in E$ if, and only if, packing piece i in plate j does not allow any other piece to be packed in j. The reason this restricted subset is enough to keep the model correctness is presented in next section.

For the enhanced formulation to have more variables than the original formulation, $|E| > |\bar{J}| + |\{x^o_{jq} : j \in J \wedge o \in O \wedge q \in Q_{jo} \wedge (o = h \wedge q > \lceil w_j/2 \rceil) \vee (o = v \wedge q > \lceil l_j/2 \rceil)\}|$ must hold, this is, the number of extraction variables must be larger than the number of pieces plus the sum of the number of cuts after the middle of each enumerated plate. Unfortunately, there is no closed formula for these sets (except $\bar{J}$ which is given), what makes necessary to compute the full enumeration to verify the difference.

3.4 The proof of correctness

The previous section presented a detailed explanation of the changes to the formulation and variable enumeration. This section proves such changes do not affect the correctness of the model. . The core of the proof consists in showing that any piece multiset that can be *packed* in some plate by the FMT formulation can also be packed by the enhanced formulation. So, below, what is meant by saying that *a plate can pack a piece multiset* is both (i) that the FMT and the enhanced formulation can represent a valid sequence of cuts obtaining such pieces from a plate but also that (ii) there is a solution for the decision problem variant of G2KP with the plate and the piece multiset as its inputs. Both meanings are essentially the same because both formulations are exact and will find a solution if it exists. The changes to FMT may be summarized to:

1. There is no variable for any cut that occurs after the middle of a plate.

2. A piece may be obtained from a plate if, and only if, the piece is the same size or smaller than the plate, and the plate cannot pack an extra piece (of any type). Obtaining a piece from a plate is always regarded as an extraction and not a cut..

The second change alone cannot affect the correctness of the model. The original formulation was even more restrictive in this aspect: a piece could only be sold if a plate

of the same dimensions existed. In the revised formulation, an extraction variable will always exist in such a case because, if a piece and plate match perfectly, there is no space for any other piece, fulfilling the only criteria for the existence of extraction variables. Consequently, what needs to be proved is that:

Theorem 3.1 (Piece extractions supersede all cuts after the middle of a plate.) *Without changing the pieces obtained from a plate, any normal cut after the middle of a plate may be replaced by a combination of piece extractions and cuts at the middle of a plate or before it.*

As mentioned before, normal guillotine cuts can be used to search for a solution (in contrast to considering a cut at every position) without loss of optimality. So the proof goes in the direction of showing that everything that can be done with normal guillotine cuts can be done without the normal cuts after the midplate but by allowing piece extractions. The theorem above and the proof below assume a plate cannot be cut twice. If a single cut is applied to a plate, then two new plates are created, and these may be further cut. There is no loss of generality by undertaking this assumption, which is already implicitly undertaken in the rest of this work. This difference can be seen as the same difference between the representation of the packing with a binary tree instead of a tree with a variable number of children.

Proof: This is a proof by exhaustion. The set of all normal cuts after the middle of a plate may be split into the following cases:

1. The cut has a perfect symmetry.
2. The cut does not have a perfect symmetry.
   (a) Its second child can pack at least one piece.
   (b) Its second child cannot pack a single piece.
      i. Its first child packs no pieces.
      ii. Its first child packs a single piece.
      iii. Its first child packs two or more pieces.

The author believes to be self-evident that the union of items 1, 2a and 2(b)i to 2(b)iii is equal to the set of all normal cuts after the middle of a plate. An individual proof for each of these cases is presented below.

Item 1 – The cut has a perfect symmetry. Two horizontal (vertical) cuts over the same plate are considered *perfectly symmetrical* if they generate the same children plate; for example, an

horizontal cut at position five of a plate of length 15 creates the same two children (of lengths five and 10) than a cut at position 10. Whether a plate is the first or second child of a cut does not make any difference for the formulation or for the problem. If the cut is in the second half of the plate, then its symmetry is in the first half of the plate. Consequently, both cuts are interchangeable, and the one after the midplate may be dismissed.

Item 2a – Its second child can pack at least one piece. Proposition 3.1 allows us to replace the second child with a size-normalised plate that can pack any demand-abiding set of pieces the original second child could pack. The second child of a cut that happens after the middle of the plate is smaller than half a plate, and its size-normalised counterpart may only be the same size or smaller. So the size-normalised plate could be cut as the first child by a normal cut in the first half of the plate. Moreover, the old first child (now second child) has stayed the same size or grown (because of the size-normalisation of its sibling), which guarantees this is possible.

Item 2(b)i – Its first child packs no piece. If both children of a single cut do not pack any pieces, then the cut may be safely ignored.

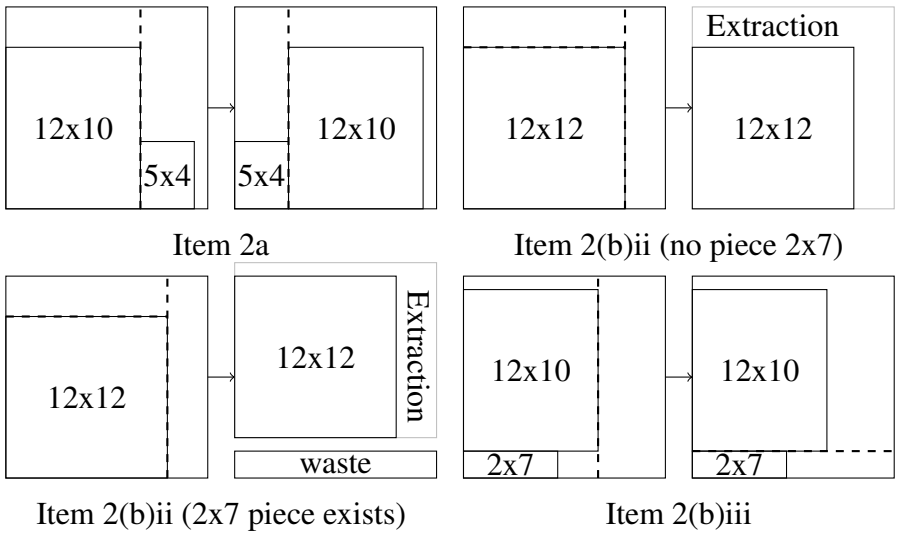
Item 2(b)ii – Its first child packs a single piece. First, let us ignore this cut for a moment and consider the plate being cut by it (i.e., the parent plate). The parent plate either: can pack an extra piece together with the piece the first child would pack, or cannot pack any extra pieces. If it cannot pack any extra pieces, this fulfils the criteria for having an extraction variable, and the piece may be obtained through it. The cut in question can then be disregarded (i.e., replaced by the use of such extraction variable). However, if it is possible to pack another piece, then there is a normal cut in the first half of the plate that would separate the two pieces, and such cut may be used to shorten the plate. This kind of normal cuts may successively shorten the plate until it is impossible to pack another piece, and the single piece that was originally packed in the first child may then be obtained by employing an extraction variable.

Item 2(b)iii – Its first child packs two or more pieces. If the first child packs two or more pieces, but the second child cannot pack a single piece (i.e., it is waste), then the cut separating the first and second child may be omitted and any cuts separating pieces inside the first child may still be done. If some of the plates obtained by such cuts need the trimming that was provided by the omitted cut, then these plates will be packing a single piece each, and they are already considered in item 2(b)ii.

Given the cases cover every cut after the middle of a plate, and each case has a proof, then follows that Theorem 3.1 is correct.

□

Figure 3.2 – Visual help for the proof of correctness



Theorem 3.1 case examples. Items 1 and 2(b)i are excluded given their simplicity. In all examples, the parent plate is 15x15. In the example of item 2a, the cut would happen after the middle of the plate, but then the pieces of the second child can be packed in the first child instead. In the example of item 2(b)ii, both cuts happen after the middle of the plate, and there are no other pieces; however, as no piece may be extracted from the leftovers, then there is an extraction variable available. In the example of item 2(b)ii, a 2x7 piece exists, but it is not extracted from the plate (the demand for it may be exhausted, for example); therefore, the extraction variable from the previous case does not exist; however, the 2x7 piece allows us to make a cut just to reduce the plate length and, for the size of the second child, an extraction variable is available. Finally, in the example of item 2(b)iii, which cut that happens first may be changed, as there is no piece packed in the subplate that would originally become the second child.

Figure 3.2 may help the reader to visualize the more complicated parts of the proof. From the cases above, the FMT formulation (from Furini, Malaguti and Thomopulos (2016)) only treats specially the pairs of cuts that are perfectly symmetrical to each other (by removing one of them).

3.5 Adaptation to the rotation variant

The adaptation of both FMT and the proposed formulation to the rotation-allowed variant of G2KP are very similar, and the author chose to employ only the proposed formulation to illustrate the process. The changes needed are:

1. change the piece set $\bar{J}$ before the call to the enumeration procedure;

2. create a new set P, which binds the two rotations of every piece;

3. change the constraint (3.12) to take into account this new set P.

The changes mentioned in item 1 consist of adding to $\bar{J}$ a new piece i' for each piece i for which $\nexists k \in \bar{J} : l_k = w_i \ \wedge \ w_k = l_i$, piece i' have $l_{i'} = w_i$, $w_{i'} = l_i$, and $u_{i'} = u_i$; differently, for each piece i for which $\exists k \in \bar{J} : l_k = w_i \ \wedge \ w_k = l_i$ (i.e., for each piece i that has its rotation as an already existing piece k), both u_i and u_k become the sum of their original values (given by the instance).

The set P mentioned in item 2 may be defined as $P = \{\{i, k\} \in P : i \in \bar{J}, k \in \bar{J}, l_k = w_i \ \wedge \ w_k = l_i\}$. Each element of P is a set of two pieces.

Finally, as mentioned in item 3, the following change is made:

$$\sum_{j \in E_{i*}} e_{ij} \leq u_i \qquad \forall i \in \bar{J} \quad (3.12)$$

to

$$\sum_{j \in E_{i*}} e_{ij} + \sum_{j \in E_{k*}} e_{kj} \leq u_i \qquad \forall \{i, k\} \in P \quad (3.15)$$

3.6 The rotation-specific mirror plate enhancement

The previous section describes only the minimal changes necessary to adapt the proposed formulation for the rotation-allowed variant. This section describes a reduction for the formulation that is only possible if rotation is allowed. This change is also compatible with the FMT formulation.

The core idea of the enhancement is that, if all pieces can rotate, then any two plates j and j', for which $l_j = w_{j'} \wedge w_j = l_{j'}$ holds, are equivalent to each other. The rationale is simple: if a set of pieces is extracted from plate j through a guillotine cutting pattern, then there exists an equivalent guillotine pattern in which every piece is rotated and which can be extracted from plate j' which is the rotation of plate j. This way, if the cut and plate enumeration generates such plates j and j', only one of them needs to be kept (but every cut that generated the removed plate needs now to generate the kept rotation instead). Finally, the opposite statement is also true (yet very inefficient): if all *plates* could somehow rotate, then the pieces themselves would not need to be able to

rotate, and distinct piece types that are rotations of each other could be considered the same.

The modifications necessary for the reduction are restricted to the cut-and-plate enumeration. Only plates in which the length is smaller than the width are allowed. If a cut would generate a plate with its length greater than its width, then the cut instead creates the rotated version of the plate, in which the length is the smallest dimension. This change can potentially reduce the number of plates (constraints) to half their original amount. Consequently, the number of variables (cuts and extractions over specific plate types) may also reduce up to half their original amount.

The only downside of mirroring the plates is that the Redundant-Cut reduction needs to be either disabled or adapted. If plate mirroring is enabled, then Redundant-Cut needs to keep track if either (or both) child plates of a cut were rotated or not. This tracking usually is not necessary to implement just the plate mirroring itself. The experiments employing rotation focus mostly on the proposed formulation (in which Redundant-Cut is already disabled because it is superseded); the few experiments with rotation-enabled FMT have Redundant-Cut disabled.

3.7 The pricing phase

The pricing procedure described in Furini, Malaguti and Thomopulos (2016), Thomopulos (2016) was reimplemented by us.]No significant changes were made to the procedure. As the experiments include multiple comparisons involving this procedure, a summary of the procedure is presented below. For simplicity, the procedure takes an already built model (from either the original formulation or the enhanced version), and any previous reductions mentioned were already applied. Clautiaux et al. (2018) refers to a similar procedure (that they apply to their own formulation) as *lagrangian filtering*; however, this term is not employed by Furini, Malaguti and Thomopulos (2016), Thomopulos (2016).

1. Fix to zero all variables representing horizontal (vertical) cuts that do not match a piece length (width).

2. Remove all integrality constraints and solve the relaxed model to obtain an upper bound for the position-only restricted problem.

3. Obtain a lower bound from an *inexact 2-staged* heuristic (FURINI; MALAGUTI;

THOMOPULOS, 2016; DOLATABADI; LODI; MONACI, 2012).

4. Employ the reduced costs of the model variables, the position-only restricted upper bound, and the heuristic lower bound to price-out variables (more details below) by fixing them to zero.

5. Restore the integrality constraints, warm-start with the heuristic solution from (step item 3), solve the model (currently, a reduced MILP model for the position-only restricted variant of the problem) and obtain a probably better lower bound. While unlikely, the heuristic may have already provided an optimal solution for the position-only restricted problem.

6. Remove all integrality constraints again.

7. **DO** solve the relaxed model, compute the reduced cost of the fixed variables, and unfix a subset of the variables with positive reduced cost **WHILE** variables with positive reduced cost exist. This loop is responsible for reintroducing any variables representing unrestricted cuts needed to solve the unrestricted variant back to the model. More details on the subset of the variables selected are below.

8. Employ the reduced costs and the upper bound, both obtained from the last solve in the loop, as well as the lower bound from the MILP solve of the position-only restricted model (item 5), to price-out variables (similarly to what was done in item 4).

9. Warm-start the model with the solution from item 5.

10. Restore integrality constraints, remove all variables yet fixed to zero, and return the model.

In item 4 and item 8, a variable is *priced out* if $\lfloor reduced\_cost(var) + ub \rfloor \leq lb$, where the upper and lower bounds are the ones available at the corresponding step. The rationale behind this requirement is straightforward. If forcing var to assume value 1 is enough to reduce the upper bound from the relaxation to less than the lower bound, then that variable (guillotine cut) cannot be used to provide a solution better than the current lower bound. Any variables necessary to produce the current lower bound are kept.

The criteria for choosing the subset of variables in each iteration of item 7 takes into account two parameters: n_{max} and $\bar{p}$. If any variables have reduced cost above $\bar{p}$, they define the subset; otherwise, the first n_{max} variables with positive reduced cost define the

subset. The original description of the procedure does mention an ordering of the variable pool, so what constitutes the *first* n_{max} *variables* is not well-defined. The author chose to interpret that the n_{max} variables of *largest reduced cost* are selected. Both parameters are automatically computed for each instance: n_{max} is one-fifth of the sum of the demand vector u, and $\bar{p}$ is one-fourth of the sum of the profits for every piece (taking demand into account).

The original description of the procedure does not indicate if, during the process, the variables are fixed and unfixed, or removed and added back. Preliminary tests indicated that the fix-and-unfix approach had better performance, so it was used in the experiments of this work. In the last step, all variables yet fixed to zero are removed.

4 EMPIRICAL ANALYSIS OF THE PROPOSED ENHANCEMENTS

There are three formulation implementations that provide data used in the comparisons of this chapter: *original* refers to the implementation presented in Furini, Malaguti and Thomopulos (2016) and in Thomopulos (2016); *faithful* refers to the reimplementation of *original* employed in this thesis; *enhanced* refers to the enhanced formulation presented in Section 3.3. The *original* implementation was not available[1]. Consequently, all data relative to *original* presented in this work comes from Thomopulos (2016). As both *original* and *faithful* refer to implementations of the FMT, the term 'FMT' is avoided in this chapter. For the sake of brevity and consistency, in this section, if a reference could be made to both Thomopulos (2016) and Furini, Malaguti and Thomopulos (2016), or to either of them, then only the former is cited. Both *faithful* and *enhanced* data were obtained by runs using the setup described in Section 4.1.

Each formulation may be modified by applying any combination of the following optional procedures: *priced* – refer to the pricing procedure described in Section 3.7 (originally from Thomopulos (2016)); *normalised* – the plate-size normalisation procedure described in Section 3.1; *warmed* – the MILP models solved were warm-started with a solution found by a previous step; *Cut-Position* and *Redundant-Cut* – are reduction procedures described in Furini, Malaguti and Thomopulos (2016) and in Thomopulos (2016), that may be enabled and disabled individually. For each experiment described in the next sections, if a procedure is not mentioned, then it is disabled. The term *restricted priced* refers to the model for the restricted version of the problem that is solved inside the pricing procedure mentioned above. Consequently, for each run of a *priced* variant, there will be a *restricted priced* run with the same combination of optional procedures. The differences between the *restricted priced* and the (unrestricted) *priced* models are mainly that: (i) the *restricted priced* model never has an horizontal (vertical) cut that does not match the length (width) of a piece; (ii) the *restricted priced* model is MIP-started with the solution of an heuristic (described in Thomopulos (2016)) while the *priced* model is MIP-started with the solution of the *restricted priced* model; (iii) the distinct solutions used to MIP-start the respective models are also used as the lower bound for the pricing procedure (details in Thomopulos (2016)).

Without the set of model variables (guillotine cuts) removed by the pricing, plates of some dimensions may become impossible to obtain. These plates are not necessary to

[1]The author os this work asked the authors of Furini, Malaguti and Thomopulos (2016) for the *original* implementation and Dimitri Thomopulos informed us it was not available.

obtain an optimal solution; otherwise, the pricing could not have removed all variables that led to them. Most of these plates could be further cut, but the value of the variables associated with such cuts can only be zero now, and, therefore, these variables can be removed too. This thinning effect may be recursive, as each newly removed variable may render some plate sizes unobtainable, similarly to what is described in Section 3.1. Hence, the pricing phase uncovers a set of unnecessary variables larger than the set it directly removes. The effort to remove such unnecessary variables and constraints is negligible. The algorithm to select which variables and constraints are kept is similar to finding the connected subgraph (starting from the original plate) in the graph representation of the formulation described at Section 3.2. In *priced* variants of *faithful* and *enhanced* this *purge* procedure is done unless stated otherwise. The experiments will show that this *purge* drastically reduces the number of variables and constraints but has almost no effect on the running times. Consequently, the author believes the solver can detect and remove such variables by itself. Nonetheless, the author encourages future comparisons to implement this *purge* procedure, as it helps determine the real size of the solved models.

Each experiment helps to substantiate choices taken in the subsequent experiments: Section 4.2 explains the choice of LP algorithms made in all remaining experiments; Section 4.3 provides evidence that *faithful* is on par with *original*, allowing us to use it as a replacement; Section 4.4 compares *faithful* to *enhanced* and shows the value of some of this thesis contributions (namely, the *normalise* procedure and the *enhanced* formulation); Section 4.5 applies the methods with best results in the last experiment to prove new optimal values and bounds for harder instances.

4.1 Setup

Every experiment in this work uses the following setup unless stated otherwise. The CPU was an AMD® Ryzen™ 9 3900X 12-Core Processor (3.8GHz, cache: L1 – 768KiB, L2 – 6 MiB, L3 – 64 MiB) and 32GiB of RAM were available (2 x Crucial Ballistix Sport Red DDR4 16GB 2.4GHz). The operating system used was Ubuntu 20.04 LTS (Linux 5.4.0-42-generic). Hyper-Threading was disabled. Each run executed on a single thread, and no runs executed simultaneously. The computer did not run any other CPU-bound tasks during the experiments. The exact version of the code used is available online (<https://github.com/henriquebecker91/GuillotineModels.jl/tree/0.2.4>), and it was run using Julia 1.4.2 (BEZANSON et al., 2017) with JuMP 0.20.1 (DUNNING;

HUCHETTE; LUBIN, 2017) and Gurobi 9.0.2 (OPTIMIZATION, 2020). The following Gurobi parameters had non-default values: `Threads` $= 1$; `Seed` $= 1$; `MIPGap` $= 10^{-6}$ (to guarantee optimality); and `TimeLimit` $= 10800$ (i.e., three hours). The next section explains the rationale for using `Method` $= 2$ (i.e., barrier) to solve the root node relaxation of the final built model; and `Method` $= 1$ (i.e., dual simplex) inside pricing (if pricing is enabled).

4.2 The choice of LP algorithm

Thomopulos (2016) do not specify the algorithm used for solving the MILP root node relaxation and, if pricing is enabled, for solving some LP models (upper bound computation) and the MILP root node relaxation of the *restricted priced* model. As Gurobi is used here, the `Method` parameter (for LP models and MILP root node relaxations) is being discussed, and not the `NodeMethod` parameter (for non-root nodes). The choice of the algorithm can drastically impact running times. A preliminary experiment included all LP algorithms available in Gurobi. Table 4.1 presents the data of the two algorithms selected for use. They are the *Dual Simplex* and the *Barrier*.

The runs use the *faithful* implementation, with *Cut-Position* and *Redundant-Cut* enabled, in its *priced* (Priced PP-G2KP in Thomopulos (2016)) and *not priced* (PP-G2KP in Thomopulos (2016)) variants. For convenience, the experiment is limited to a few instances. This subset consists of all instances for which the *Complete PP-G2KP Model* finds the optimal solution within the time limit in Furini, Malaguti and Thomopulos (2016) (Table 2). If pricing is disabled, the root node relaxation contributes to most of the running time. This characteristic makes them a good choice for this experiment.

The following conclusions can be derived from Table 4.1. Using the *Barrier* algorithm in the pricing phase is not viable. This impracticality happens because the pricing phase includes an iterative variable pricing phase. This iterative phase repeatedly adds variables to one LP model and solves it again. The *Barrier* algorithm solves every LP from scratch; the *Dual Simplex* reuses the previous basis and saves considerable effort. However, *Barrier* performs better if there is no previous base to reuse. Consequently, the configuration chosen was *Dual Simplex* for the pricing phase and *Barrier* for the root relaxation of the final model.

Table 4.1 – Comparison of LP-solving algorithms used inside solving procedure

Instance	Dual Simplex			Barrier			DS + B
	N. P.	R. %	Priced	N. P.	R. %	Priced	Priced
CU1	27.37	92.11	3.79	24.18	94.68	3040.82	**3.58**
STS4	93.49	89.88	48.80	49.94	77.32	7851.30	**47.75**
STS4s	103.20	94.92	39.29	43.74	86.34	8470.41	**38.36**
gcut9	226.68	72.29	**3.92**	51.48	85.77	2060.04	4.01
okp1	51.95	84.18	38.89	**32.41**	67.78	–	38.79
okp4	98.25	93.35	144.30	**72.09**	92.31	–	141.53
okp5	178.13	89.89	252.09	**96.38**	67.24	–	239.44

Dual Simplex and *Barrier* indicate the respective algorithm was used for all LPs and root node relaxations, *DS + B* means that *Dual Simplex* was used to solve all LPs inside the pricing phase, and *Barrier* was used to solve the root node relaxation of the final model. The columns *N.P.* (*Not Priced*) and *Priced* display the time to solve (in seconds) using the aforementioned variant. The columns *R.%* refer to the per cent of the time spent by *Not Priced* in the root node relaxation of the final model. Source: the author.

4.3 Comparison of *faithful* against *original*

Without a reimplementation of *original*, any comparison would need to be made directly against the data in Thomopulos (2016). However, such a comparison would hardly be fair, as it compares across machines, solvers, and programming languages. Also, for example, it does not allow us to assess the benefits of applying the *plate-size normalisation* procedure to the *original* formulation. The purpose of this section is to show that *faithful* may be fairly used in place of *original*. For this purpose, Table 4.2 compares the number of model variables and number of plates of the diverse model variants presented in Thomopulos (2016). The chosen dataset is, therefore, the same as the one used in these works for the comparison to be possible. The dataset aggregates 59 instances of the previous literature from many distinct sources, and all instances are either artificially generated or of undisclosed origin. A detailed entry about this dataset and all of its constituting instances can be found in appendix A under the FMT59 (which is the name adopted for this dataset in this work). The number of enumerated plates strongly correlates to the number of constraints in the model. Thomopulos (2016) presents the number of plates, not the number of constraints. To simplify the comparison, the same is done here.

The *Priced PP-G2KP* runs in Thomopulos (2016) had three time limits of one hour to solve: the restricted model (i.e., obtaining a lower bound); the iterative variable pricing (i.e., obtaining an upper bound); the final model. Such configuration always generates a final model. However, it also has two drawbacks:(i) the computer performance

Table 4.2 – Comparison of *faithful* against *original*

Variant	T. L.	E. R.	O. #v	F. %v	O. #p	F. %p
Complete PP-G2KP	0	0	156,553,107	100.00	1,882,693	100.00
Complete +Cut-Position	0	0	103,503,930	99.99	1,738,263	100.01
Complete +Redundant-Cut	0	0	121,009,381	109.94	1,882,693	100.00
PP-G2KP (CP + RC)	0	0	74,052,541	120.05	1,738,263	100.01
Restricted PP-G2KP	0	0	5,335,976	99.28	306,673	99.99
Priced Restricted PP-G2KP	0	1	3,904,683	102.20	305,690	99.99
(no purge) Priced PP-G2KP	3	7	14,619,460	93.74	1,642,382	100.01
Priced PP-G2KP	3	7	14,619,460	31.92	1,642,382	25.55

The sum of columns *T. L.* (Time Limit) and *E. R.* (Early Return) gives the number of instances excluded from consideration in the respective row. Column *T. L.* has the number of instances for which *faithful* reached the time limit without generating the respective model variant – these instances are: Hchl7s, okp2, and okp3. The column *E. R.* has the number of instances for which this thesis reimplementation found an optimal solution before generating the respective model variant. Columns *O. #v* and *O. #p* refer to *original*.Column *O. #v* (*O. #p*) presents the sum of variables (plates) for the instances in which *faithful* generated a model. Columns *F. %v* and *F. %p* refer to *faithful* Column *R. %v* (*R. %p*) has the sum of variables (plates) in the generated models as a percentage of the quantity obtained by the original implementation. Source: the author.

may define the answer given in the first two phases, affecting the size of the final model (and making it harder to make a fair comparison);(ii) if the restricted model, or the iterated variable pricing, cannot be done in one hour, then the final model will probably hit the time limit too – in Thomopulos (2016), every run that hits one of the two first time limits also hits the third time limit. The author chose to use a single three-hour time limit for the experiments of this chapter. In the other chapters, either the more common one-hour time limit is employed or, for relatively short experiments, no time limit is employed.

Table 4.2 references the names used in Thomopulos (2016). The *Complete PP-G2KP* is the formulation with all optional procedures disabled, while the *PP-G2KP* mean both *Cut-Position* and *Redundant-Cut* are enabled. *Restricted PP-G2KP* and its priced version are solved inside *Priced PP-G2KP* runs. If the lower and upper bounds found during pricing are the same, then the optimal solution was found before generating the final model. The instances in which this happened for an unrestricted solution are 3s, A1s, CU1, CU2, W, cgcut1, and wang20. The instance A1s presented this behaviour already in the pricing of the restricted model.

The *original* had no *purge* phase after pricing. Consequently, for the columns that refer to *original*, the last row just repeats the data of the row above.

The following conclusions can be derived from Table 4.2. All variants, except *Priced PP-G2KP*, are within $\pm 0.01\%$ of the expected number of plates (and, conse-

quently, of constraints). The *Complete PP-G2KP*, *Complete +Cut-Position*, and *Restricted PP-G2KP* are within $\pm 1\%$ of the expected number of variables. The number of variables in both *Complete +Redundant-Cut* and *PP-G2KP (CP + RC)* is $10 \sim 20\%$ larger than expected. Given the experiments isolate such divergence to cases in which Redundant-Cut is enabled, the author believes there is some disagreement between the original implementation of Redundant-Cut and its reimplementation. However, the reimplementation follows closely the description given in Thomopulos (2016). The number of variables and plates in *Priced* variants is not entirely deterministic. The number of variables of *Priced* variants is either slightly above ($+2\%$) or lower ($-6 \sim 68\%$).

For all non-*priced* variants, the fraction of the running time spent in the model generation is negligible. Consequently, the comparison presented in Table 4.2 is sufficient. The author cannot say the same for the *priced* variants. Thomopulos (2016) does not report the size of the multiple LP models solved inside the iterative pricing (a phase of the pricing). For instances in which *original* and *faithful* executed all phases of pricing and solved the final model, the *original* spent 34.35% of its time in the iterative pricing phase, while *faithful* spent 61.69%. It is hard to pinpoint the source of this discrepancy. One possible explanation is that, in *original*, other phases took more time than they took in *faithful*. For example, *faithful* uses the *barrier* algorithm for the root node relaxation of the final model, which reduces the percentage of time spent in this phase. Nevertheless, for the subset of the instances aforementioned, the total time spent by *faithful* was about 13% of the time spent by *original*. While the difference between machines and solvers does not allow us to infer much from that figure, the author believes that the magnitude of the difference guarantees that *faithful* is not a gross misrepresentation.

4.4 Comparison of *faithful* against *enhanced*

The primary purpose of this section is to evaluate the impact of the proposed enhancements to the state of the art. The contributions evaluated here are the *normalise* reduction (i.e., the plate-size normalisation presented in Section 3.1) and the *enhanced* formulation (presented in Section 3.3.1). The state of the art consists in a formulation (*Complete PP-G2KP*), two reductions (*Cut-Position* and *Redundant-Cut*), and a pricing procedure presented in Furini, Malaguti and Thomopulos (2016), Thomopulos (2016). In this section, the reimplementation of *Complete PP-G2KP* named *faithful* (to distinguish from the data of the *original*) is employed. The author also reimplemented the reductions

Table 4.3 – Comparison of *faithful* vs. *enhanced* over the 59 instances used in Thomopulos (2016)

Variant	T. T.	#e	#m	#s	#b	S. T. T.	#variables	#plates
Faithful	106,057	–	59	53	0	41,257	88,901,964	1,738,366
Enhanced	25,538	–	59	58	2	14,738	3,216,774	231,836
F. +Normalizing	60,078	–	59	56	0	27,678	60,316,964	610,402
E. +Normalizing	14,169	–	59	59	52	14,169	2,733,125	145,157
F. +N. +Warming	60,542	–	59	56	0	28,142	60,316,964	610,402
E. +N. +Warming	9,778	–	59	59	4	9,778	2,733,125	145,157
Priced F. +N. +W.	49,919	8	50	55	0	6,719	3,210,857	174,214
Priced E. +N. +W.	9,108	8	51	59	1	9,108	600,778	64,904
P. F. +N. +W. -Purge	50,054	8	50	55	0	6,854	8,072,810	544,892
P. E. +N. +W. -Purge	9,209	8	51	59	0	9,209	1,021,526	134,102

The meaning of each column follows: *T. T.* (Total Time) – sum of the time spent in all instances including timeouts, in seconds; *#e* (early) – number of instances in which pricing found an optimal solution (and, consequently, did not generate a final model); *#m* (modeled) – number of instances that generated a final model; *#s* (solved) – number of solved instances; *#b* (best) – number of instances that the respective variant solved faster than any other variant; *S. T. T.* (Solved Total Time) – same as Total Time but excluding runs ended by time or memory limit; *#variables* (*#plates*) – sum of the variables (plates) in all generated final models (see column *#m*). The first row (Faithful) has two runs that ended in memory exhaustion. The time of these runs is accounted for as they were timeouts. Source: the author.

and the pricing procedure, but as *enhanced* may also enable these optional procedures, the text avoids labelling them as *faithful* to minimise confusion.

The *faithful* and *enhanced* formulations cannot be combined. However, both allow enabling any combination of the optional procedures. The only exception is *Redundant-Cut*, which is unnecessary for *enhanced* and, therefore, never applied to it. Outside of this exception, in this section, *Redundant-Cut* and *Cut-Position* are always enabled. These reductions never increase the number of variables (or constraints), cost negligible computational effort, and were already discussed in Furini, Malaguti and Thomopulos (2016), Thomopulos (2016).

This section also discusses the effects of the *purge* procedure and warm-starting the non-*priced* model. The deterministic heuristic used to MIP-start the non-*priced* models is the same used in the *restricted priced* model solved inside the pricing procedure.

Considering the data from Table 4.3, the following statements can be made:

1. *enhanced* solves more instances than *faithful* (using at most 24% of its time);
2. the number of variables of ‘Enhanced’ is almost the same as ‘Priced F. +N. +W.’;
3. between ‘Enhanced’ and ‘Priced F. +N. +W.’ the former has better results;

4. *normalise* further reduces variables by $14 \sim 32\%$ and plates by $37 \sim 65\%$;
5. MIP-starting *enhanced* makes it slightly slower in 52 instances;
6. MIP-starting *enhanced* saves more than one hour in the other 7 instances;
7. any benefit from MIP-start in 'F. +N. +Warming' was negated by its timeouts;
8. *purge* greatly reduces the model size but has almost no effect on running time;
9. the effects of applying *pricing* to *enhanced* are not much better than *purge*;
10. applying *pricing* to *faithful* is positive overall but loses one solved instance.

Both the number of variables (cuts) and plates (constraints) are reduced by *enhanced*. The reduction in the number of variables (cuts) is a direct consequence of the *enhanced* differential: making unnecessary any cuts after the middle of each plate. However, the reduction in the number of plates (constraints) is an *indirect* consequence of the same differential.

One of the ways the *enhanced* reduces the number of constraints is by innately avoiding the creation of some size-normalised plate types. The length (width) of an horizontal (vertical) cut is always normalised, i.e., a demand-abiding combination of piece lengths (widths), and so is the length (width) of the first child, but there is no guarantee about the length (width) of the second child. If the cut happens after the middle of the plate and the length (width) of the second child is not normalised (or the second child cannot pack any piece and is discarded as waste), then there is no perfectly symmetrical cut in the first half of the plate. Horizontal (vertical) cuts in the first half of the same plate (the only ones available to the *enhanced* formulation) cannot create a second child with the same normalised length (width) of the previously mentioned first child. Therefore, plates with the non-normalised length (width) are obtained by both formulations, but the normalised counterparts are obtained only by the *original* formulation. The fact that a plate is normalised (or not) is irrelevant in itself (at least for the considered formulations). The fact that the *enhanced* formulation has the non-normalised plate instead of its normalised counterpart is not relevant to the performance of the formulation (much less to its correctness). However, having both the non-normalised plate and the normalised counterpart increases the number of constraints without any clear advantage and, therefore, negatively impacts the performance of the *original* formulation.

Considering the data from Table 4.4, the following statements can be made:

Table 4.4 – Fraction of the total time spent in each step (only runs that executed all steps)

Variant	Time	E %	H %	RP %	IP %	FP %	LP %	BB %
Priced Faithful +N. +W.	6,632	0.12	0.38	26.16	57.36	2.91	4.56	8.29
Priced Enhanced +N. +W.	1,178	0.03	2.18	50.89	23.66	0.46	2.70	19.95
P. F. +N. +W. -Purge	6,766	0.11	0.37	26.00	57.03	2.81	5.12	8.45
P. E. +N. +W. -Purge	1,185	0.03	2.18	50.70	23.64	0.46	2.83	20.09

Time is the sum of all time (in seconds) spent in the 47 instances that finished all phases in all four variants considered. These are the same 47 indicated in row *Priced F. +N. +W.* of Table 4.3. From the 59 instances dataset, 4 had timeout (Hchl4s, Hchl7s, okp2, and okp3), and 8 found an optimal solution inside pricing (3s, A1s, CU1, CU2, W, cgcut1, okp4, and wang20). All remaining columns present percentages of the time spent in a specific phase: *E* – enumeration of cuts and plates (and all reductions); *H* – restricted heuristic used to warm-start the restricted priced model; *RP* – restricted pricing (not including the heuristic time); *IP* – iterative pricing; *FP* – final pricing; *LP* – root node relaxation of the final model; *BB* – branch-and-bound over the final model. Source: the author.

1. both *BB* and *LP* phases are slightly faster with *purge* as expected;

2. both *E* and *H* phases are almost negligible (at most 2% with *H* in *enhanced*);

3. together the *RP* and *IP* phases account for $74.5 \sim 83.5\%$;

4. *RP* and *IP* swap percentages between *enhanced* and *faithful*;

5. *faithful* shows some overhead in all phases strongly affected by model size.

4.5 Evaluating *enhanced* against harder G2KP instances

The purposes of the experiment described in this section are: (i) to show the limitations of the *enhanced* formulation against more challenging instances; (ii) to provide better bounds and new proven optimal values for such instances.

Velasco and Uchoa (2019) proposes a set of 80 hard instances to test the limitations of their bounding procedures; these instances are employed in this section. The instances were artificially generated and are divided into four classes of 20 instances each. The dataset focuses on two characteristics: (i) the area of the pieces is small compared to the area of the original plate (the average ratio varies between 1.6% and 5%); (ii) each class is defined by the shape of the original plate, and the likely shape of the randomly generated pieces. The original plates of the first two classes have one dimension two or four times larger than the other dimension. In the first class, the pieces are likely to be larger in the same dimension the original plate is larger; in the second class, the pieces are likely to

Table 4.5 – Summary table for the instances proposed in Velasco and Uchoa (2019)

C.	Variant	#m	Avg. #v	Avg. #p	T. T.	#s	Avg. S. T.
1	Not Priced	20	1,787,864.55	22,316.50	172,574	5	2,114.85
	Restricted Priced	13	467,692.15	17,139.00	180,051	5	3,610.29
	Priced	5	264,315.80	11,978.40	196,733	3	4,377.77
2	Not Priced	20	1,533,490.70	18,638.50	167,973	5	1,194.68
	Restricted Priced	20	453,159.70	18,638.30	155,184	8	3,198.11
	Priced	8	394,613.88	9,735.50	178,812	4	1,503.01
3	Not Priced	20	2,895,300.75	33,249.40	171,155	5	1,831.11
	Restricted Priced	10	431,913.00	15,895.80	174,569	5	2,513.80
	Priced	5	372,597.00	13,287.80	179,712	4	1,728.08
4	Not Priced	20	3,201,374.45	35,197.10	167,776	7	3,910.89
	Restricted Priced	10	497,802.20	17,011.00	197,047	2	1,323.65
	Priced	2	211,093.00	14,227.00	199,477	2	2,538.79

Summary table for the instances proposed in (VELASCO; UCHOA, 2019). The columns are: *C.* – instance class (described in (VELASCO; UCHOA, 2019), 20 instances each); *Variant* – the solving method employed; *#m* (modeled) – number of instances in which the model was built before timeout; *Avg. #v* and *Avg. #p* – the average number of variables and plates in the *#m* instances that generated a final model for the respective variant; *T. T.* (Total Time) – sum of the time spent in all instances in seconds, including timeouts; *#s* (solved) – number of instances solved; *Avg. S. T.* (Avg. Solved Time) – as total time but excludes timeouts and divides by *#s*. Averages were used instead of simple sums because the very different number of generated and solved models made the sums misleading. Source: the author.

be larger in the dimension the original plate is shorter. The original plates of the last two classes are squares. The pieces of the third class have, on average, the same dimension with double the size of the other; in the fourth class, half of the pieces follow the previous distribution, and the other half invert the favoured dimension. More details can be found in the Appendix.

Only two variants were executed for this experiment, the *priced* and non-*priced* versions of *enhanced* with *Cut-Position*, *normalise*, and *MIP-start* enabled. The results for the *restricted priced* variant are also presented because this variant is solved inside a step of the *priced* variant (the same reductions apply to it). Table 4.5 presents a summary of all runs, and Table 4.6 to 4.9 presents the improved bounds and solved instances.

For this experiment, Gurobi was allowed to use the 12 physical cores of the employed machine. Gurobi distributes the effort of the branch-and-bound (B&B) phase equally among all cores. Solving an LP (as a root node relaxation, or not) calls barrier, primal simplex, and dual simplex. Each of the simplex methods uses a single thread, while barrier uses all remaining cores, and Gurobi stops when the first of them finishes.

Concerning the data from Table 4.5, the author wants to highlight some unexpected results: (i) the total number of instances solved by the *restricted priced* was slightly smaller than non-*priced*, even with non-*priced* solving the harder *unrestricted* problem; (ii) many runs reached time limit without solving the continuous relaxation of the *restricted* model (necessary for creating *restricted priced* model); (iii) non-*priced* solved more instances than *priced* in all cases. It is worth noting that the *priced* variant could have been considered the best configuration in the previous dataset, as its total time was shorter than non-*priced* (both solved all instances). Ideally, the pricing procedure would significantly reduce the size of the model and, consequently, the root node relaxation and B&B phases would take much less time to solve. However, the gain in decreasing the size of the (already reduced) *enhanced* model further does not seem to compensate for the cost of solving hard LPs more than once. Also, previous sections have shown that reducing the model size does not guarantee that the running time will be reduced by the same magnitude.

The purpose of Table 4.6 to 4.9 is to allow querying the exact values for specific instances. Even so, there are some gaps to fill. For the instances presented in Table 4.6 to 4.9, the min / mean / max gap between the heuristic lower bound and the final lower bound were: 0.38 / 18.08 / 37.03 (non-*priced*); 0.68 / 20.62 / 37.29 (*restricted priced*); 9.17 / 19.38 / 32.24 (*priced*). In other words, no solution, or best bound, was given by the heuristic, and most of the time, its solution was considerably improved. For the reader convenience, it can be said that this experiment has: proved 22 unrestricted optimal values (5 already proven by Velasco and Uchoa (2019), confirming their results); proved 22 *position-only* restricted optimal values (in an overlapping but distinct subset of the instances); improved lower bounds for 25 instances; improved upper bounds for 58 instances.

Table 4.6 – V&U instances either solved (restricted or unrestricted) or with improved bounds. (PART I)

Instance	Lower Bounds for Unrestricted				RP UB	Upper Bounds for Unr.		
	RP	P	NP	V&U		P	NP	V&U
P1_100_200_25_1	**27,251**	**27,251**	**27,251**	**27,251**	27,251	**27,251**	**27,251**	27,340
P1_100_200_25_2	**25,090**	**25,090**	**25,090**	24,870	25,090	25,403	**25,389**	25,522
P1_100_200_25_3	**25,730**	**25,730**	**25,730**	**25,730**	25,730	25,974	**25,909**	26,088
P1_100_200_25_4	26,732	**26,896**	**26,896**	26,769	26,732	**26,896**	**26,896**	27,051
P1_100_200_25_5	**26,152**	–	**26,152**	25,772	26,565	–	**26,617**	26,857
P1_100_200_50_1	28,388	–	**28,440**	28,388	28,504	–	**28,440**	28,558
P1_100_200_50_2	**26,276**	**26,276**	**26,276**	**26,276**	26,276	**26,276**	**26,276**	26,326
P1_100_200_50_3	**27,192**	–	**27,192**	27,165	27,536	–	**27,483**	27,679
P1_100_200_50_4	28,058	–	**28,095**	27,977	28,345	–	**28,340**	28,388
P1_100_200_50_5	**27,722**	–	**27,722**	27,603	27,930	–	**27,722**	28,009
P1_100_400_25_1	53,247	–	53,008	**53,904**	54,540	–	**54,707**	55,038
P1_100_400_25_2	–	–	41,275	**44,581**	–	–	**47,091**	47,097
P1_100_400_25_3	42,748	–	46,222	**47,455**	✻	–	**49,371**	49,473
P1_100_400_25_4	–	–	38,567	**40,517**	–	–	**46,069**	46,078
P1_100_400_25_5	44,482	–	**53,220**	53,205	✻	–	54,120	**54,063**
P1_100_400_50_1	–	–	53,831	**55,856**	–	–	**56,897**	57,074
P1_100_400_50_2	–	–	40,440	**48,373**	–	–	**51,754**	51,893
P1_100_400_50_4	–	–	**55,107**	52,708	–	–	**55,654**	55,661
P1_100_400_50_5	–	–	**53,749**	53,502	–	–	**55,005**	55,454

Instances solved (position-only restricted or unrestricted) or with improved bounds. Lower and upper bounds that are valid for the unrestricted problem are grouped. Column *RP UB* (restricted priced upper bound) is kept separate as it is not a valid bound for the unrestricted problem. Bold indicates the best unrestricted bounds for the instance. If the LB and the UB are the same for the same instance and variant, both values are underlined. The instance names follow the pattern `Class_L_W_n_seed`. The sub-headers mean: *RP* – Restricted Priced (solved inside *P* runs); *P* – Priced; *NP* – Not Priced; *V&U* – obtained by Velasco and Uchoa in (VELASCO; UCHOA, 2019).

✻ These runs hit the time limit at the very start of the upper bound computation and, consequently, they produced only large and irrelevant upper bounds, which the author chose to omit to keep the table formatting.

Table 4.7 – V&U instances either solved (restricted or unrestricted) or with improved bounds. (PART II)

Instance	Lower Bounds for Unrestricted				RP UB	Upper Bounds for Unr.		
	RP	P	NP	V&U		P	NP	V&U
P2_200_100_25_1	**21,494**	**21,494**	**21,494**	**21,494**	21,494	**21,494**	**21,494**	**21,494**
P2_200_100_25_2	25,244	**25,413**	**25,413**	**25,413**	25,244	**25,413**	**25,413**	25,648
P2_200_100_25_3	25,282	**25,397**	**25,397**	**25,397**	25,282	**25,640**	25,647	25,723
P2_200_100_25_4	25,729	–	**25,734**	25,437	26,181	–	**26,239**	26,898
P2_200_100_25_5	26,211	**26,413**	**26,413**	26,220	26,211	26,728	**26,413**	26,898
P2_200_100_50_1	**25,679**	–	25,626	25,627	26,233	–	**26,282**	26,447
P2_200_100_50_2	**27,801**	**27,801**	**27,801**	27,789	27,801	**27,801**	**27,801**	27,943
P2_200_100_50_3	27,435	**27,453**	**27,453**	**27,453**	27,435	27,584	**27,579**	27,596
P2_200_100_50_4	27,395	–	**27,439**	27,362	27,668	–	**27,704**	27,718
P2_200_100_50_5	**29,386**	**29,386**	**29,386**	**29,386**	29,386	**29,386**	**29,386**	**29,386**
P2_400_100_25_1	49,327	–	**49,947**	49,026	50,218	–	**50,365**	51,006
P2_400_100_25_2	48,312	–	**48,542**	47,773	49,268	–	**49,315**	49,908
P2_400_100_25_3	**46,970**	–	46,860	45,406	47,113	–	**47,204**	48,938
P2_400_100_25_4	**51,051**	–	49,847	49,521	51,526	–	**51,600**	52,229
P2_400_100_25_5	**49,620**	–	48,832	47,403	50,440	–	**50,580**	54,248
P2_400_100_50_1	54,550	54,550	**54,679**	52,890	54,550	54,981	**54,916**	55,629
P2_400_100_50_2	**54,821**	–	54,768	53,492	55,183	–	**55,181**	55,543
P2_400_100_50_3	54,141	–	**54,747**	54,216	55,537	–	**55,709**	56,065
P2_400_100_50_4	53,375	–	**54,240**	48,649	54,857	–	**54,987**	55,604
P2_400_100_50_5	**53,763**	–	53,541	50,047	54,893	–	**54,918**	55,471

Table organization is the same as Table 4.6. Source: the author.

Table 4.8 – V&U instances either solved (restricted or unrestricted) or with improved bounds. (PART III)

Instance	Lower Bounds for Unrestricted				RP UB	Upper Bounds for Unr.		
	RP	P	NP	V&U		P	NP	V&U
P3_150_150_25_1	29,896	**29,989**	**29,989**	29,896	29,896	**29,989**	**29,989**	30,005
P3_150_150_25_2	**29,345**	–	29,196	29,101	29,906	–	29,965	**29,961**
P3_150_150_25_3	**30,286**	**30,286**	**30,286**	**30,286**	30,286	**30,286**	**30,286**	30,327
P3_150_150_25_5	**31,332**	**31,332**	**31,332**	30,924	31,332	31,715	**31,682**	31,839
P3_150_150_50_1	31,377	**31,701**	**31,701**	**31,701**	31,377	**31,701**	**31,701**	31,892
P3_150_150_50_2	30,846	–	**30,884**	**30,884**	31,110	–	**31,008**	31,115
P3_150_150_50_3	32,037	**32,121**	**32,121**	32,050	32,037	**32,121**	**32,121**	32,240
P3_150_150_50_4	**31,925**	–	**31,925**	**31,925**	32,210	–	**31,925**	32,070
P3_150_150_50_5	**31,631**	–	31,521	31,448	31,857	–	**31,896**	31,901
P3_250_250_25_1	–	–	51,027	**58,480**	–	–	**60,548**	60,611
P3_250_250_25_2	–	–	63,646	**68,070**	–	–	**73,316**	73,339
P3_250_250_50_1	–	–	59,072	**67,603**	–	–	**76,117**	76,341
P3_250_250_50_2	–	–	62,772	**75,569**	–	–	**82,644**	82,666

Table organization is the same as Table 4.6. Source: the author.

Table 4.9 – V&U instances either solved (restricted or unrestricted) or with improved bounds. (PART IV)

Instance	Lower Bounds for Unrestricted				RP UB	Upper Bounds for Unr.		
	RP	P	NP	V&U		P	NP	V&U
P4_150_150_25_1	30,870	–	**30,923**	**30,923**	31,094	–	**30,923**	31,130
P4_150_150_25_2	30,576	–	**30,687**	30,460	30,786	–	**30,687**	30,931
P4_150_150_25_3	30,257	–	**30,352**	**30,352**	30,501	–	**30,352**	**30,352**
P4_150_150_25_4	30,055	**30,106**	**30,106**	**30,106**	30,055	**30,106**	**30,106**	**30,106**
P4_150_150_25_5	**30,582**	–	30,102	**30,582**	30,952	–	**31,228**	31,286
P4_150_150_50_1	**31,673**	**31,673**	**31,673**	**31,673**	31,673	**31,673**	**31,673**	**31,673**
P4_150_150_50_2	32,302	–	**32,317**	**32,317**	32,434	–	**32,317**	32,423
P4_150_150_50_3	30,906	–	**30,913**	30,882	31,500	–	**31,519**	31,756
P4_150_150_50_4	31,912	–	**31,961**	31,912	32,206	–	**31,961**	32,140
P4_150_150_50_5	**32,027**	–	31,845	31,864	32,331	–	**32,308**	32,484
P4_250_250_25_4	–	–	69,530	**79,476**	–	–	**81,634**	81,839
P4_250_250_50_2	–	–	67,675	**77,206**	–	–	**87,314**	87,331
P4_250_250_50_4	–	–	69,063	**78,359**	–	–	**86,941**	87,069

Table organization is the same as Table 4.6. Source: the author.

5 COMPARISON TO OTHER FORMULATIONS OF THE LITERATURE

This chapter compares the proposed formulation (BBA) to another five formulations of the recent literature besides its immediate predecessor (FMT). A concise review of these other formulations is given in Section 5.1. Differently from BBA and FMT, the employed implementation of these formulations was written in C++ for the CPLEX solver and not by the author of this thesis. To control differences between the implementations, the design of the experiments is different from the other chapters: the implementations are used only to generate and save the models into files (not to solve the models), and both CPLEX and Gurobi are called from the command line to solve each of these saved models. To avoid any doubts about the specific experiment setup and design, the same is exhaustively described in Section 5.2 and Section 5.3. Besides the comparison between the formulations in Section 5.5, a comparison between solvers is also presented (Section 5.4), which allows us to dispel any doubts about the choice of solver favouring one formulation over another.

Another relevant distinction is that these other formulation are, in general, more compact and easier to implement in a GAMS-like framework. As it was seen in Section 4.4, the time spent by FMT/BBA in the cut-and-plate enumeration is often negligible (less than 0.2% for the FMT59 dataset), however implementing this enumeration correctly and efficiently is no trivial task.

5.1 Concise review of the newly considered formulations

For the sake of explanation, the author chose to aggregate some of the formulations in the same paragraph when they share similar modelling strategies. The author seeks to highlight how the interpretation of solutions can lead to very different formulations.

The BCE formulation, proposed for the Guillotine Strip Packing Problem in Messaoud, Chu and Espinouse (2008) and adapted for the G2KP in Martin et al. (2020b), is based on a theorem that characterizes guillotine patterns and uses coordinates at which items may be located. The theorem states that a pattern is of guillotine type if, and only if, for any region (i.e., sub-rectangle) of the object, at least one of the following conditions is satisfied: (i) this region contains only a single item; (ii) the segments of the piece length in this region on the x-axis consist of at least two disjoint intervals; and, (iii) the segments of the piece width in this region on the y-axis consist of at least two disjoint intervals. The

formulation is compact in the numbers of variables and constraints with $O(n^4)$ for the GSPP, where n is the number of pieces to be packed. This formulation seems to recall the interval-graph approach of Fekete and Schepers (1997) for the non-guillotine Orthogonal Packing Problem.

The MLB formulation, proposed in Martin et al. (2020b), assumes that each solution can be represented by a sequence of horizontal and vertical guillotine cuts over a two-dimensional grid interpretation of object $L \times W$. It was inspired by a formulation for the non-guillotine G2KP from Beasley (1985b). In a MLB model, a binary variable x_{kij} represents the allocation of the left-bottom corner of an piece type $k \in \{1, \ldots, m\}$ to a point (i, j) on the object, $0 \leq i \leq L - l_k$, $0 \leq j \leq W - w_k$. Taking into consideration the constraints from Beasley (1985b), it ensures a constrained pattern and avoids the overlap between any pair of allocated/cut pieces, which is related to a maximum clique problem. Then it satisfies the guillotine cutting with binary variables for horizontal cuts $h_{ii\prime j}$, $0 \leq i < i\prime \leq L$, $0 \leq j \leq W$, vertical cuts $v_{ijj\prime}$, $0 \leq i \leq L$, $0 \leq j < j\prime \leq W$, and enabled rectangles $p_{i_1 i_2 j_1 j_2}$, $0 \leq i_1 < i_2 \leq L$, $0 \leq j_1 < j_2 \leq W$. The main concepts involve associating: (i) the variables x_{kij}, $h_{ijj\prime}$ and $v_{ii\prime j}$ by prohibiting horizontal and vertical cuts on allocated pieces and imposing the allocation of the pieces on cut corners; (ii) the variables $h_{ii\prime j}$, $v_{ijj\prime}$ and $p_{i_1 i_2 j_1 j_2}$ by allowing only horizontal and vertical edge-to-edge cuts in enabled rectangles. The formulation is pseudo-polynomial in the numbers of variables and constraints with $O(mLW + L^2W^2)$. As expected, one can reduce the number of variables and constraints by using the discretization of normal sets or related ones Herz (1972), Christofides and Whitlock (1977).

The MM1 and MM2 formulations, proposed in Martin, Morabito and Munari (2020a), are inspired in the bottom-up strategy of successive horizontal and vertical builds of the pieces. A build envelops two small rectangles to generate a larger rectangle. For instance, as introduced in Wang (1983), the horizontal build of pieces $l_1 \times w_1$ and $l_2 \times w_2$ provides a larger rectangle of size $(l_1 + l_2) \times \max\{w_1, w_2\}$, and the vertical build provides a larger rectangle of size $\max\{l_{1,2}\} \times (w_1 + w_2)$. Defining an MM1 or MM2 model, it requires to previously determine an upper bound $\bar{n}$ to the maximum number of builds on object $L \times W$ (e.g., $\bar{n} = \sum_{i \in I} b_i$). The MM1 formulation is pseudo-polynomial as its definition requires an explicit binary tree structure, which is generated by a procedure that considers upper bound $\bar{n}$ as an input. This binary tree structure is represented by a set of triplets (j, j^-, j^+), where j^- and j^+ are the left and right child nodes of node j, respectively; the root node $j = 1$ represents the object (i.e., the solution). Its main concepts

involve ensuring: (i) each node j of the binary tree structure can represent either a copy of an item type $i \in I$ (binary variable z_{ji}), an horizontal build (binary variable x_{jh}), a vertical build (variable x_{jv}), or it is not necessary in the solution; (ii) the solution represents a guillotine pattern (i.e., a virtual binary tree) by linking the variables x_{jo}, $o \in \{h, v\}$, of each parent node j with the variables of its child nodes j^- and j^+; and, (iii) the variables L_j and W_j are considered to represent, respectively, the length and width of a node j according to the previous definition of horizontal and vertical builds, over variables z_{ji} and x_{jo}. The MM2 formulation, however, is compact as it considers the set of binary variables y_{jk}, $j, k \in \{1, \ldots, \bar{n} - 1\}$, $j < k$, for representing implicitly the binary tree structure. The MM1 and MM2 formulations were first proposed as integer non-linear programs, and then they were linearized through the use of disjunctive inequalities of big-M type.

The MM3 formulation, proposed in Martin, Morabito and Munari (2020b), is inspired in the top-down strategy of successive cuts on the original and residual objects towards the items. It makes use of the binary tree structure initially proposed for the MM1 formulation. As a consequence, it presumes the same constraints for representing a guillotine pattern (i.e., a virtual binary tree) by linking the variables x_{jo}, $o \in \{h, v\}$, of each parent node j with the variables of its child nodes j^- and j^+. However, its geometric constraints are non-trivial. Since variables L_j and W_j are no longer in the formulation, the sizes of the residual objects (i.e., nodes of the binary tree structure) are defined according to the decisions taken in the previous residual objects. Alternatively stated, the decisions of each node j take into consideration the previous decisions of all its ancestral nodes up to the root node through disjunctive inequalities of big-M type.

5.2 Experiments setup

Every experiment in this section used the following setup. The CPU was an AMD® Ryzen™ 9 3900X 12-Core Processor and 32GiB of RAM were available. The operating system was Ubuntu 20.04 LTS (Linux 5.4.0). Two kernel parameters had non-default values: `overcommit_memory = 2` and `overcommit_ratio = 95`. Hyper-Threading was disabled. Each run executed on a single thread, and no runs executed simultaneously. The computer did not run any other CPU bound task during the experiments.

The models for the BBA and FMT formulations were built using the Julia language and the Gurobi solver. The models for the BCE, MLB, MM1, MM2, and MM3

formulations were built using C++ and the CPLEX solver. To homogenize the experiments, these implementations were used only to build the models and then save them to MPS files. Each selected combination of formulation, rotation configuration, and instance originated a single MPS file. A Julia script then executed each MPS file in four different configurations: CPLEX/LP, CPLEX/MILP, Gurobi/LP, and Gurobi/MILP.

The implementation of BBA and FMT formulations is available at an online repository[1]. The scripts for (i) saving the BBA and FMT models as MPS and (ii) solving all MPS files are also availables[2]. The implementations of all the other formulations, as well as the script for generating the MPS files, are available upon request to Martin Pereira Martin[3] who graciously let the author borrow his implementations for the purpose of this comparison. The same version of the compilers and solvers was used for the MPS generation and the MPS solving phases.Those are: Julia 1.5.3, g++ 9.3.0, CPLEX 20.1, and Gurobi 9.1.1.

In both CPLEX and Gurobi some non-default configurations were used. The solvers were configured to: employ a single thread; use a specified seed (`CPX_PARAM_RANDOMSEED`, in CPLEX, and `Seed`, in Gurobi, were set to one); employ an integer tolerance adequate for the instances; avoid finishing with suboptimal solutions for the selected datasets (`CPX_PARAM_EPGAP`, in CPLEX, and `MIPGap`, in Gurobi, were set to 10^{-6}); and respect an one hour time-limit (`CPX_PARAM_TILIM`, in CPLEX, and `TimeLimit`, in Gurobi, were set to 3600). For a more graceful handling of memory exhaustion, set CPLEX parameter `CPXPARAM_MIP_Limits_TreeMemory` is set to 28672 (Gurobi does not seem to provide a similar parameter). Only when solving the FMT and BBA formulations, the solver employs the barrier method for solving the LP and for solving the root node relaxation (`CPXPARAM_LPMethod` and `CPXPARAM_MIP_Strategy_StartAlgorithm`, in CPLEX, were both set to 4, and `Method`, in Gurobi, was set to 2).

5.3 Outline of the experiments

A short description of the instance datasets used in this section follows. More details about each dataset can be found in the Appendix.

[1] See <https://github.com/henriquebecker91/GuillotineModels.jl/tree/0.5.0>

[2] See <https://github.com/henriquebecker91/phd/tree/BMC-1>

[3] The ORCID of Martin Pereira Martin is 0000-0002-6722-7571. He is the main author of Martin et al. (2020b), Martin, Morabito and Munari (2020a), and Martin, Morabito and Munari (2020b).

CU/CW Datasets introduced by Fayard, Hifi and Zissimopoulos (1998). Their names stand for Constrained (demand) and Unweighted/Weighted. They totalise 22 instances: CU1–11 and CW1–11.

APT Dataset introduced by Alvarez-Valdés, Parajón and Tamarit (2002). The whole dataset consists of 40 instances (APT10–49), however, only the second half (APT30–49) is employed here, because the first half is for the unconstrained demand variant. The APT30–39 are unweighted and APT40–49 are weighted.

FMT59 Group of instances assembled by (FURINI; MALAGUTI; THOMOPULOS, 2016) with instance subsets from previous datasets. Already employed in previous chapters.

Easy18 A subset of FMT59 defined by the author for this section. Its purpose is to reduce the number of runs needed before discarding a formulation from further consideration. The dataset contains: cgcut1–3, gcut1–12, OF1–2, and wang20.

The author selected these datasets because the prior work already employed them. For the CU, CW, and APT datasets, with and without rotation, the best known lower bounds from Velasco and Uchoa (2019) are used. For the FMT59 dataset, without rotation, Furini, Malaguti and Thomopulos (2016) presents every optimal value[4], but there is no comprehensive source on the best known values for this dataset when rotation is allowed.

Each run can be uniquely identified by a combination of instance, formulation, rotation configuration (allow rotation or not), solve mode (MILP or LP), and solver (CPLEX or Gurobi). The first three characteristics determine an MPS file; the last two determine four distinct runs over the same MPS file.

The whole set of runs consists of:

1. The Easy18 instances combined with each of the seven considered formulations and both rotation configurations, except by the BCE formulation with rotation enabled, which was not implemented.

2. The CU, CW, and FMT59 instances combined with the BBA, MM3, hierachical, and MM2 formulations and both rotation configurations.

[4]There is only one typo: the optimal value of the okp2 instance is 22502, not 22503.

3. The APT instances combined with the four formulations mentioned above but only with rotation disabled. No runs found optimality with rotation disabled and, therefore, the author decided to not spend computational effort in the rotation-enabled counterparts.

5.4 Comparison between CPLEX and Gurobi

This section aims to answer two questions: (i) is one of the solvers superior in this context? (ii) does a choice of solver benefit a specific formulation?

Table 5.1 answers the first question by revealing a small but consistent advantage for the Gurobi solver. Nevertheless, Gurobi does not completely dominate CPLEX, as each solver had some instances only solved by it.

Table 5.1 – Comparison amongst CPLEX and Gurobi results.

Solver	Type	#opt	#u. opt	#best	#c. best	Avg. T (s)	Avg. S. T. (s)
CPLEX	MILP	288	12	96	16	2435.86	455.21
Gurobi	MILP	302	26	218	63	2339.84	353.62
CPLEX	LP	704	4	194	6	379.17	40.62
Gurobi	LP	720	20	530	24	297.79	31.78

The meaning of each column follow: *#opt* – number of runs finished by optimality; *#u. opt* – number of optimal runs unique to the respective solver (i.e., other solver did not reach optimality); *#best* – number of optimal runs in which the respective solver finished before the other solver (counting the ones not finished by the other solver); *#c. best* – number of clean best times, i.e., optimal runs that took at least one minute for the respective solver and either were not solved by the other solver or it took double the time to solve; *Avg. T. (s)* – mean run time in seconds (runs ended by timeout or memory exhaustion are counted as taking one hour); *Avg. S. T. (s)* – mean run time of solved runs in seconds.

Table 5.2 – Comparison amongst CPLEX and Gurobi results by formulation.

Measure	BCE	BBA	FMT	MLB	MM1	MM2	MM3
Optimal	105.88	99.31	131.57	150.00	100.00	100.00	101.24
T. Time	85.45	101.75	74.71	58.61	45.09	27.68	67.60

Percentage of solved runs and total time spent by Gurobi in relation to CPLEX, broken down by formulation, for all MILP runs. Runs ended by time or memory limit are counted as taking one hour. Source: the author.

Table 5.2 answers the second question. In the first row, figures above 100% mean Gurobi solved more runs than CPLEX and, in the second row, figures below 100% mean Gurobi spent less time than CPLEX. Gurobi has better results for all formulations except

the BBA formulation, in which the results are very similar (only slightly worse). The choice of Gurobi as a solver improves the results for some formulations more than others, but, in general, the formulations which solve fewer instances are the most beneficial. Therefore, the author considers Gurobi a fair choice for the rest of the paper

5.5 Comparison between formulations

This section aims to provide empirical evidence for the choice of one formulation over another and to identify the impact of allowing rotation over all formulations. Given the number of considered formulations, Table 5.3 filters the considered formulations further.

Table 5.3 – Filtering formulations with EASY18 dataset.

	Fixed					Rotation				
Method	#opt	g_{lb}	Avg. T.	g_{ub}	#f	#opt	g_{lb}	Avg. T.	g_{ub}	#f
BCE	2	7.00	3341	7.31	0	–	–	–	–	–
BBA	18	0.00	< 1	1.74	0	18	0.00	< 1	0.63	0
FMT	13	27.78	1336	2.48	4	10	44.44	1723	0.71	7
MLB	10	33.62	1671	3.85	4	3	78.23	3002	3.22	9
MM1	16	0.08	750	7.31	0	14	0.31	1082	3.32	0
MM2	10	0.33	1684	7.31	0	9	0.45	1885	3.32	0
MM3	16	0.07	685	7.31	0	12	0.43	1297	3.32	0

The explanation of these columns follows: #opt – the number of runs finished by optimality; g_{lb} – the average percentage gap between the best lower bound found and the best known lower bound (if the run finishes without a solution, as is the case of memory exhaustion, it is assumed that a trivial empty solution was returned); Avg. T. – the average total time spent by a run in seconds (both timeout and memory exhaustion count as one hour); g_{ub} – the average percentage gap between the continuous relaxation and the best known lower bound; #f – the number of runs finished by timeout or memory exhaustion during the root node relaxation phase (these are excluded from g_{ub}). Source: the author.

Table 5.3 shows that, for the EASY18 dataset, BBA dominates all other formulations. The MLB has the largest average lower bound gap. The model size often prevents its runs from finishing solving the root node relaxation. The same problem is also seen in FMT runs but to a smaller extent. BCE solves the least instances; its lower bound gap is smaller than FMT and MLB but considerably above the rest of the instances. MM1 and MM3 solve most instances and have very small lower bound gaps. Finally, MM2 solves a number of instances comparable to FMT and MLB but, different from them, the root node relaxation is always solved, and a good primal solution is delivered.

Table 5.4 – Solving datasets CU and CW

	CU							
	Fixed				Rotation			
Alg.	#opt	g_{lb}	Avg. T.	g_{ub}	#opt	g_{lb}	Avg. T.	g_{ub}
BBA	10	9.09	425	0.21	9	18.18	716	0.06
MM1	3	0.54	2928	1.45	0	0.68	3600	0.57
MM2	0	0.80	3600	1.45	0	0.88	3600	0.57
MM3	3	0.78	3021	1.45	2	0.97	3400	0.57
	CW							
	Fixed				Rotation			
BBA	11	0.00	15	1.24	10	0.00	496	1.72
MM1	5	0.00	2560	11.13	3	0.01	3052	5.26
MM2	0	0.89	3600	11.13	0	0.51	3600	5.26
MM3	5	0.00	2602	11.13	2	0.80	3259	5.26

The columns are the same as of Table 5.3 except #f is ommited because no run was interrupted in the middle of solving the root node. Source: the author.

Considering these results, the authors chose to remove BCE, MLB, and FMT from further comparison. The rationale for these choices follows: BCE solves very few instances leading to a great increase in experiment times; the model size of MLB leads to memory problems, especially for runs allowing rotation; and FMT is similar to BBA but without some additional enhancements.

In Table 5.4, it can be seen that two distinct behaviours emerge. The BBA (pseudo-polynomial) starts to present a behaviour similar to FMT: either solving the instances faster than the other formulations, or failing to solve the root node relaxation at all[5]. The other three formulations have difficulty proving optimality; however, they always solve the root node relaxation and provide primal solutions of good quality. The g_{ub} column indicates that MM1, MM2, and MM3 have the same average upper bound gap. The reason for this similarity is that the three formulations, while distinct, use the same additional constraints to tighten the upper bound to a precomputed value. In all three formulations, these constraints impose a tighter bound than the one imposed by the remainder of the formulation, leading to this similarity. The problem becomes harder for all formulations if rotation is allowed. The values in the g_{ub} column for MM1, MM2, and MM3 reduce when rotation is allowed; however, this only happens because their upper bounds stay the same while the best known solution increases in value.

[5]The table omits it but BBA fails to solve the root node relaxation one time for CU/Fixed and two times for CU/Rotation.

Table 5.5 – Solving datasets FMT59 and APT

	FMT59									
	Fixed					Rotation				
Method	#opt	g_{lb}	Avg. T.	g_{ub}	#f	#opt	g_{lb}	Avg. T.	g_{ub}	#f
BBA	57	1.69	183	1.75	1	56	1.05	233	3.28	1
MM1	27	1.00	2387	4.89	0	20	-1.16	2657	4.66	0
MM2	10	1.40	3015	4.89	0	9	-1.04	3077	4.66	0
MM3	25	1.39	2328	4.89	0	13	-0.74	2837	4.66	0
	APT									
	Fixed					Rotation				
BBA	0	100.00	3600	–	20	–	–	–	–	–
MM1	0	11.32	3601	1.86	0	–	–	–	–	–
MM2	0	3.10	3600	1.86	0	–	–	–	–	–
MM3	0	90.39	3509	1.90	9	–	–	–	–	–

The columns are the same as of Table 5.3.

Table 5.5 corroborates the findings of Table 5.4. BBA solves more FMT59 instances but ends up with a larger g_{lb} than the other formulations because of the poor solution quality in the few unsolved instances. For the FMT59 instances, MM1 has the lowest g_{lb} but, for the APT instances MM2 surpasses it. The BBA cannot solve the root node relaxation for any APT instances during the one-hour time limit. The column FMT59/Rotation/g_{lb} has negative values because, as mentioned in Section 5.3, the author chose to use the known optima from fixed orientation for this particular dataset.

6 HYBRIDISATION WITH THE RESTRICTED FORMULATION

This chapter proposes another symmetry-breaking change compatible with the formulations considered in Section 3.3, this is, FMT and the proposed formulation (BBA). This change further complicates the formulation, and the empirical results did not reveal an improvement as large as the previously discussed enhancements. Therefore, the author chose to keep this change self-contained in this chapter. The author is unaware of any previous application of the proposed change to unrestricted 2D guillotine problems. The Cut-Position enhancement from Furini, Malaguti and Thomopulos (2016) draws inspiration from the same broad idea: to get closer to a formulation for the (simpler) restricted problem while keeping optimality for the unrestricted problem. However, the proposed change and the Cut-Position both approach this goal in distinct and complementary ways.

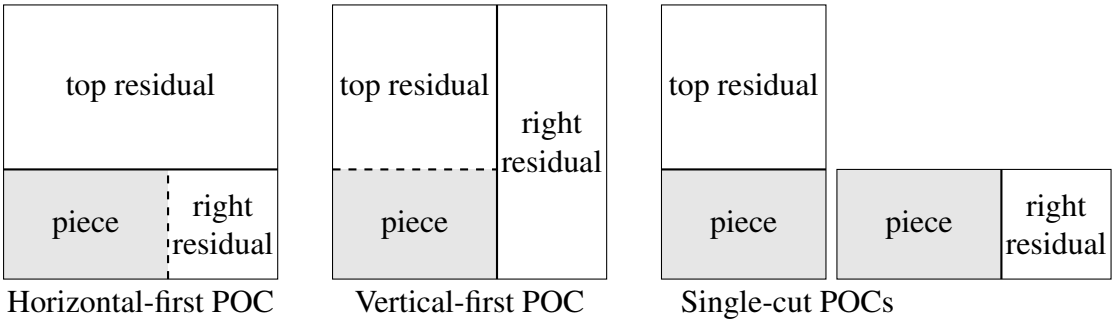
6.1 The restricted problem and piece-outlining cuts

A guillotine cutting problem is said to be *restricted* if (i) each horizontal (vertical) guillotine cut must match the length (width) of a piece that fits into the plate, i.e., it happens at a *restricted cut position*, and (ii) a piece of that length (width) is guaranteed to be obtained from the first child plate. The concept of a *restricted* variant appears first in the context of the three-staged guillotine cutting problem. The two-staged problem is inherently restricted: a cut that does not match the outline of a piece, or a cut that does not guarantee a piece extraction because it is not paired with a cut from the only other stage, is a cut that will not help to obtain any pieces before the two stages are over. Only when the number of stages is three or more that an optimal solution for the unrestricted problem may require cuts without such immediate purposes. Applying the concept of *restricted* to unlimited stages is not new, Furini, Malaguti and Thomopulos (2016) already does it. Furini, Malaguti and Thomopulos (2016) also presents an intermediary variant which respects (i) but not (ii), this variant can be referred to as *position-only* restricted problem. The *position-only restricted problem* is the one solved by the *restricted priced* in Chapter 4.

The restricted problem has at least two performance advantages over the unrestricted problem. The first advantage is related to the number of restricted cut positions: the number of cuts positions in any plate is bounded by the number of pieces (i.e., linear on the input) and not pseudo-polynomial (i.e., bounded by plate dimensions), even if the

number of plates themselves is still pseudo-polynomial. The second advantage is related to the piece extraction requirement. There is no optimality loss if, after a cut at a restricted position related to a single piece, it is immediately determined that, if necessary, the first child plate will be cut again in the next stage to obtain the respective piece. The possibility of joining two decision variables together has led previous prior on the restricted problem, as Silva, Alvelos and Carvalho (2010), to redefine *cut* to mean *one or two guillotine cuts associated a priori to a piece type and which outline and obtain a piece-sized plate that cannot be further cut*. The guillotine cuts considered until now may incidentally outline and obtain a piece-sized plate as their child plates. However, they are not a priori associated with a single piece type, nor do they guarantee their first child plate (if piece-sized) cannot be further cut. In this chapter, the text distinguishes between these two kinds of cuts to avoid confusion. The single and unassociated cuts considered until now will be referred to as *basic guillotine cuts* (or BGCs for short), and this new definition of cut will be referred to as *piece-outlining cuts* (or POCs for short). Figure 6.1 may help to visualise the *piece-outlining cuts*

Figure 6.1 – Piece-outlining cuts



Souce: the author.

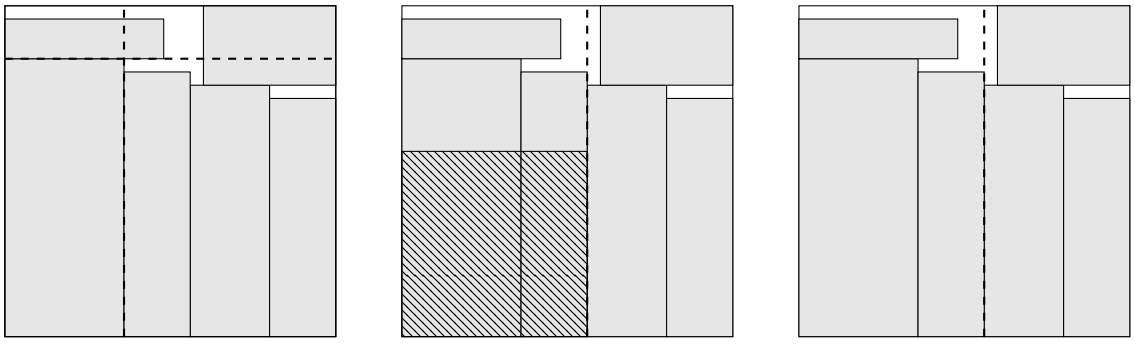
While a POC constituted by two BGCs may be considered a single decision by a solving method and may be seen as happening in succession, in practice, stage restrictions may change the order a cutting machine performs them. However, these real-world details do not impact the modelling and will not be discussed in this chapter. Essentially, each piece type that fits into a plate has two POCs associated with it. One POC that does the horizontal guillotine cut first and then obtains the piece from the first child plate through a vertical cut (if necessary). This POC always leaves a *top residual plate* (second child plate of the first cut) and often a *right residual plate* (second child plate of the second cut). The other POC is the same, except that the vertical cut is done first (i.e., always leaving a right residual and often a top residual plate). Finally, the piece-sized plate obtained by a

POC is the first child plate of the second cut if the second cut exists; otherwise, just the first child plate of the only cut. The piece-sized plate is either immediately regarded as an obtained piece (already enforcing a rule of the restricted problem) or may be considered waste (e.g., the cutting stock problem often allows piece overproduction). However, the piece-sized plate is *never* treated as an intermediary plate that could be further cut.

A caveat of the coupled representation mentioned above is that, for some instances of the restricted problem, the number of POCs may be larger than the number of restricted cut positions. In general, each piece type that fits into a plate has two POCs[1] (vertical-first and horizontal-first). An horizontal (vertical) BGC at a restricted position is shared by all piece types with the same length (width). However, the main advantage of the coupled representation comes from breaking symmetries, not reducing the number of variables.

The POCs are a natural choice for the *restricted* problem but not for the *unrestricted* problem for mostly two main reasons. The first reason is that, in the restricted problem, each horizontal (vertical) cutting position shares length (width) with at least one piece. However, in the unrestricted problem, some cutting positions can only be reached by combining many pieces. The second reason is that the definition of the *restricted* problem guarantees that employing only POCs cannot lead to optimality loss; the same is not true for the unrestricted problem (see Figure 6.2).

Figure 6.2 – Distinctions between, restricted, position-only restricted, and unrestricted problems.



The restricted problem cannot obtain the unrestricted optimal solution. If the first cut happens at a restricted position, the child plates cannot fit the six pieces of the optimal solution, regardless of the piece chosen to be obtained first from the original plate and the orientation of the first cut employed. The position-only restricted problem can obtain the unrestricted optimal solution if, by chance, there is an unpacked piece with a width that matches the necessary vertical cut; otherwise, the solution is also out of reach. Source: the author.

[1]The exception happens when the piece type shares the length or the width with the plate and, consequently, both POCs are equivalent and can be considered the same.

Silva, Alvelos and Carvalho (2010) proposes a mathematical formulation for the two-stage and three-stage restricted cutting stock problems. The formulation was not named by its authors; hence, in this text, it will be referred to as SAV (from the author's surname initials: Silva, Alvelos, and Valério). The SAV is very similar to the FMT, which is examined in Section 3.2. In fact, the SAV may be seen as an FMT variant that uses POCs instead of BGCs. The limitation to two- and three-stage problems comes from the cut-and-plate enumeration. If the enumeration is not stopped at a specific stage, the SAV immediately supports unlimited stages. Essentially, the proposed change is to: hybridise the FMT with the SAV, replacing BGCs with POCs *only* when doing so cannot lead to loss of optimality for the unrestricted problem.

6.2 Implementation details

As seen in the last section, a POC (*piece-oulining cut*) is prefered over a BGC (*basic guillotine cut*) if it is guaranteed that replacing the latter by the former will not cause loss of optimality. For the restricted problem, the typical set of horizontal (vertical) cutting positions is just the set of unique values in l_i (w_i) for every piece type i that fits into the plate. Besides one corner case, each single guillotine cut at such positions may be replaced by the corresponding POC. The corner case arises in cutting positions that come from a length (or width) value shared by two or more pieces. In this case, a single guillotine cut needs to be replaced by two or more POCs, depending on how many pieces share the corresponding cutting position; otherwise, the model would lose the capability to produce that piece type.

For the unrestricted problem, the exact set of cutting positions often varies between different solving methods. There are many discretisation procedures (see Section 3.3) and reductions to be applied either after or during such discretisations. The author will focus on the discretisations and reductions procedures employed by the formulations of Chapter 4 (this is, the FMT and BBA formulations). The base discretisation employed by both FMT and BBA is straightforward: q is an horizontal (vertical) cutting position if, and only if, there is a demand-abiding linear combination of lengths (widths) from pieces that fit into the respective plate. This cutting position set is a superset of the restricted set (from the last paragraph) and will be referred to as the *base unrestricted set*. Suppose a cutting position allows for the associated BGC to be replaced by (one or more) POCs without loss of optimality for the *unrestricted problem*. In that case, the cutting

position (and, by extension, the BGC) is said to be *replaceable*.

A cutting position must meet two conditions to be deemed replaceable. The first condition is that a cutting position of the same orientation for the same plate exists in the restricted set. This first condition is necessary because, otherwise, the cut is not outlining a piece, i.e., there is no corresponding piece type to be extracted from the first child plate. The second condition is that such horizontal (vertical) cutting positions cannot be obtainable by a demand-abiding linear combination of two or more piece lengths (widths), considering only the pieces that fit into the respective plate. This second condition is necessary because, otherwise, the replaced cut could be necessary for the only optimal cutting pattern of an instance of the unrestricted problem. An example of this situation can be seen in Figure 6.2 (the middle pattern, i.e., Position-only Restricted). The middle vertical cut matches a piece width (i.e., it satisfied the first condition); however, if it were replaced by a POC associated with the square piece, it would be impossible to obtain the unrestricted optimal solution (that needs a BGC at the same position).

The two reductions proposed in Furini, Malaguti and Thomopulos (2016), *Cut-Position* and *Redundant-Cut*, cause little change to the replaceable cutting positions. Both reductions are briefly described at the start of Section 3.3.2. The only cutting positions removed by *Cut-Position* are the ones not in the restricted set and, therefore, not replaceable. Moreover, if the cutting position set of a plate is reduced by *Cut-Position* and the kept positions are all replaced with POCs, then that plate and any plate strictly smaller than it will, in fact, be solved by the SAV formulation instead of the FMT formulation. *Redundant-Cut* may remove a replaceable cutting position. However, the predicted alternative cutting position from a larger plate will always be replaceable too, and replacing it with one or more POCs never requires adding back the cuts removed by Redundant-Cut. Also, the BBA formulation never has trim cuts like those removed by Redundant-Cut (see Section 3.3), so this enhancement is superseded by it.

BBA adds extraction variables and reduces the base unrestricted set to only the cutting positions up to the midplate. The extraction variables can be seen as POCs in which both top and right residual plates are guaranteed to be waste; therefore, extractions are not subject to be replaced by POCs. BBA requires us to differentiate between *binding* and *non-binding* POCs. A POC is *non-binding* if the piece-sized plate it obtains may be regarded as waste; conversely, if the piece-sized plate must be sold as a piece, then the POC is *binding*. A *binding* POC cannot be employed if an extra copy of the associated piece type would lead to disrespecting the demand constraint. If replaceable cuts in the

BBA formulation are replaced by *binding* POCs, then there are cases in which loss of optimality occurs. The cause of this loss of optimality is that, in BBA, a replaceable cut may be required by an optimal solution even if there is no demand for the associated piece. These seemingly unnecessary cuts aim to reduce the plate size until a large piece can be obtained from the plate through an extraction variable. A complete example follows.

Example 6.1 (Hybridised BBA with binding cuts loses optimality.) *Consider the following G2KP instance:* $L = 100$, $W = 100$, $l = [100, 100]$, $w = [1, 51]$, $u = [1, 1]$, *and* $p = [1, 1]$. *The optimal solution clearly must contain the only available copy of each of the two piece types. In BBA, there is no cut after the midplate; consequently, a vertical cut at position* 51 *is ruled out. The only possibility is a vertical cut at position* 1 *for which the first child plate could be immediately sold as the single copy of the first piece type. The second child plate (*$100x99$*) also does not have an extraction variable for the immediate extraction of the second piece type (*$100x51$*). The BBA determines that for an extraction variable to exist "[...] the plate cannot fit an extra piece (of any type)." and the first piece type fits together with the second in the* $100x99$ *plate. Again, a vertical cut at position* 51 *is unavailable because it happens after midplate. Consequently, BBA forces the optimal solution to create* 50 *plates of size* $100x1$*, one of which will be sold as a piece, and the rest considered waste. The second child of the* 50*th (and last) cut has size* $100x50$*, and it can be sold as the second piece type because an extraction variable is now available (i.e., the previously quoted condition does not apply anymore). The adoption of* binding *POCs makes it impossible for BBA to obtain an optimal solution for this example. The reason is that there are not* 50 *copies of the first piece type, but these would be needed by the* 50 *binding piece-outlining cuts necessary to obtain an optimal solution. The same problem does not arise if the POCs are not binding.*

The corner case of two or more pieces sharing the same length/width needs to be considered in the unrestricted problem too, but with a subtle distinction. In the restricted problem, replacing every single guillotine cut by POCs also brings the advantage of not needing an additional mechanism to enforce the problem definition (i.e., to guarantee piece extractions from the first child plates). However, in the unrestricted problem, the choice between replacing a single guillotine cut by multiple POCs, or keeping it as a guillotine cut, is just a trade-off between model size and model symmetry. Therefore, this work further distinguishes between two implementations of hybridisation. The *conservative* hybridisation substitutes each replaceable horizontal (vertical) BGC with

one horizontal-first (vertical-first) POC that is associated with the single piece type that matches the length (width) of the cutting position (and that fits into the respective plate). If two or more fitting piece types match the cutting position, the conservative hybridisation leaves the BGC unchanged. The *aggressive* hybridisation substitutes each replaceable horizontal (vertical) BGC with one horizontal-first (vertical-first) POC *for each piece type* that matches its length (width) (and that fits into the respective plate).

The author believes it is excessive to present the full formulation and implementation details for every combination of the FMT/BBA formulation with conservative/aggressive hybridisation and binding/non-binding POCs. The experiments in the next section only consider the BBA with conservative/aggressive hybridisation and non-binding POCs. The distinction between conservative and aggressive hybridisation is mostly made at the cut and plate enumeration; however, because of an unfortunate notation detail explained further, it is less troublesome to present an accurate formulation of the conservative hybridisation than the aggressive hybridisation. In light of this, the author chose to fully present the conservative hybridised BBA formulation with non-binding POCs. The main differences in implementing other combinations are briefly discussed shortly after.

The *conservative hybridised BBA formulation with non-binding* POCs requires a new set of variables, a new set of constraints, a new parameter, and some minor changes to the objective function and some of the existing constraints. Both the new set of variables and the new set of constraints are bounded by $|\bar{J}|$ and, therefore, cause only a small relative increase to the model size of a non-trivial instance. The notation for the new variable and parameter set follows:

$s_i \ \forall i \in \bar{J}$ – Integer variable. Indicates how many piece-sized plates obtained by POCs associated with piece type i were sold as pieces of type i. By *sold* the author means they contributed to the objective function and were accounted for by the demand constraint.

$h^o_{qji} \ \forall o \in O, j \in J, q \in Q_{jo}, i \in \bar{J}$ – Binary parameter. Byproduct of the cut and plate enumeration. It has value one if cut x^o_{qj} is a POC that produces a piece-sized plate corresponding to piece i; zero otherwise.

Some variables, parameters, and constraints need just a little reinterpretation or no change at all. The already established parameter a^o_{qkj} is exactly the same for BGCs and has a slightly different meaning for POCs. The difference is that the j (obtained

child plate) is always either the top or right residual (i.e., the POC version of the first and second child) and that both o (orientation) and q (cutting position) refer only to the first constituting cut of a POC; the meaning of k (parent plate) is left unchanged. The set of variables representing cuts (x^o_{qj}) also does not need change, as h^o_{qji} fills the need to identify POCs and their associated piece types. Consequently, the constraints (3.10) and (3.11) presented below are the same as the non-hybridised formulation.

The constraint (6.2) guarantees each piece-sized plate available (s_i) comes from an actual POC. The remaining changes consist into adding s_i to the demand constraint (6.3) (which avoids overproduction without prohibiting the POCs themselves) and to the objective function (6.1) (which allows piece-sized plates to be sold).

$$\boldsymbol{max.} \sum_{(i,j)\in E} p_i e_{ij} + \sum_{i\in\bar{J}} p_i s_i \tag{6.1}$$

$$\boldsymbol{s.t.} \sum_{o\in O}\sum_{q\in Q_{jo}} x^o_{qj} + \sum_{i\in E_{*j}} e_{ij} \leq \sum_{k\in J}\sum_{o\in O}\sum_{q\in Q_{ko}} a^o_{qkj} x^o_{qk} \qquad \forall j \in J, j \neq 0, \tag{3.10}$$

$$\sum_{o\in O}\sum_{q\in Q_{0o}} x^o_{q0} + \sum_{i\in E_{*0}} e_{i0} \leq 1 \qquad , \tag{3.11}$$

$$s_i \leq \sum_{j\in J}\sum_{o\in O}\sum_{q\in Q_{jo}} h^o_{qji} x^o_{qj} \qquad \forall i \in \bar{J}, \tag{6.2}$$

$$s_i + \sum_{j\in E_{i*}} e_{ij} \leq u_i \qquad \forall i \in \bar{J}, \tag{6.3}$$

$$x^o_{qj} \in \mathbb{N}^0 \qquad \forall j \in J, o \in O, q \in Q_{jo}, \tag{3.13}$$

$$e_{ij} \in \mathbb{N}^0 \qquad \forall (i,j) \in E \tag{3.14}$$

$$s_i \in \mathbb{N}^0 \qquad \forall i \in \bar{J}. \tag{6.4}$$

The aforementioned unfortunate notation detail is the incapability of denoting two or more different cuts x^o_{qj} with the same orientation o and the same cutting position q over the same plate j. Therefore, if the aggressive hybridisation replaces a BGC with two or more POCs, then the notation does not allow us to differentiate between them.The a^o_{qkj} parameter also needs to change, as it suffers from the same problem. The aggressive hybridisation code deals with this problem by having unique single indexes for each cut and reverse indexes from each cut property (like orientation or cutting position) to the cuts themselves; this way, the cuts are not limited to the uniqueness of some property combination.

A trivial way to change the presented formulation to use *binding cuts* is to change

the constraint set (6.2) to require equality. However, the binding cuts can also be implemented without the new variable and constraint sets. The term s_i could just be replaced by $\sum_{j \in J} \sum_{o \in O} \sum_{q \in Q_{jo}} h^o_{qji} x^o_{qj}$ in both the objective function and the demand constraint. Both mentioned ways to implement binding cuts work on the FMT formulation, which does not have the same loss of optimality problem as the BBA.

6.3 Experimental results

In these experiments, for reasons explained further ahead, each instance was solved ten times with ten distinct solver seeds. The BBA configuration included all applicable reductions previously discussed (i.e., Cut-Position and Plate-Size Normalisation) but excluded initialisation with a primal heuristic and pricing. The barrier algorithm was used to solve the root node as usual. Only the Gurobi solver is used in these experiments. No runs ended in timeout. The computer setup, as well as the Julia and Gurobi versions/parameters, are the same as described in Section 5.2, but the model was built and solved in the same process (i.e., there was no writing and reading from MPS file), and no time limit was enforced. Three variants are scrutinised: no hybridisation (N. H.), conservative hybridisation (C. H., avoids increasing model size), and aggressive hybridisation (A. H., always hybridise, even if it leads to an increase of the model size). The first dataset considered is FMT59 (solved as G2KP), and the second is CJCM (solved as G2OPP); more details on these datasets can be found in appendix A.

Table 6.1 shows that both C. H. and A. H. had a similar impact on the total solving time (i.e., a reduction of $\approx$20%). A. H. had slightly better timings despite the considerable increase in the number of cuts. C. H. slightly reduces the number of cuts. Both C. H. and A. H. have almost no effect on the number of plates (or extractions variables). The percentage of hybridised cuts (h %) and hybridised cuts with just one residual (k %) show that the new reductions changed a very significant part of the models. The number of instances with the lowest averages shows that N. H. is the best option for most instances.

A closer look into the data, see Table 6.2, reveals that most time difference comes from a few hard instances. In fact, the instance Hchl4s alone is responsible for most of the difference, with okp2 having about half its relevance and the rest of the instances considerably less impact. The number of variables hybridised (H columns) does not seem a good indicator of how impacted the solving times will be. However, if N. H. spends most of the time solving the root node (low Non-Root %), C. H. and A. H. generally

Table 6.1 – Summary of hybridisation impact over BBA formulation and FMT59 dataset.

Variant	T. T.	Δ B. T.	#b	#extr.	#cuts	h %	k %	#plates
N. H.	6,681	1,703	27	186,536	2,498,801	–	–	113,822
C. H.	5,468	489	22	184,067	2,496,421	41	18	113,373
A. H.	5,447	469	10	184,050	3,021,911	67	28	113,366

T. T. (Total Time) – sum of the mean time of all instances, in seconds; Δ *B. T.* (Distance from the Best Time) – sum of the difference in mean time between the respective variant and the variant with the lowest mean time for the same instance, in seconds, i.e., if all variants ran in parallel and had average time, how much time the runs of the respective variant would spend after another thread already finished; *#b* (best) – number of instances in which the respective variant had the lowest (best) average time among the variants; *#extr.* – total number of extraction variables (considering one model per instance); *#cuts.* – total number of cut variables (considering one model per instance); *h %* – percentage of #cuts that were hybridised; *k %* – percentage of #cuts that were not only hybridised but also discarded the second child of the second constituting cut as waste, i.e., the POC resulted in the piece-sized plate and *one* other plate; *#plates* – total number of plates (considering one model per instance). Source: the author.

do not bring great time improvements. As the most significant reductions often occur in instances that spend less than 1% of the time in the root node, the time distribution does not change significantly. An exception is CHL1s which shows that C. H. seems to impact not the time at the root node but the time at the B&B, as expected from a symmetry breaking-enhancement.

The coefficient of variation of the analysed instances reveals the reason for multiple runs with distinct seeds: the difference between two runs of the same variant but distinct seeds is often larger than the difference between the means of two distinct variants. Intuitively, breaking symmetries should reduce the variance of the timings. By cutting symmetric branches, there is less opportunity for a solver seed to traverse multiple equivalent branches with a good relaxation (but bad primal) before finding a primal solution that cuts all such branches. In fact, when C. H. and A. H. achieve a considerable (20% or more) reduction of the mean time, the coefficient of variation (which is relative to the mean time) generally shows a reduction. However, a more general effect, i.e., a higher percentage of model hybridisation (H%) leading to lower CV (or mean time), is not observed. Exactly *which* variables were hybridised probably have more impact than *how many* variables. Finally, the reduction of variance, while positive if the objective is to compare solution methods, may be unwanted when solving the same problem in parallel. For example, if two methods have similar mean times, the method with the most variance will probably have a thread find the optimal solution first.

Table 6.2 – Impact of BBA hybridisation in FMT59 instances taking more than 10s.

	H (%)		Mean Time (s/%)			CV (%)			Non-Root (%)		
Inst.	C	A	N (s)	C (%)	A (%)	N	C	A	N	C	A
Hchl4s	46	57	3,657	80	**69**	76	35	25	>99	>99	>99
okp2	22	22	1,844	**77**	88	21	19	44	>99	>99	>99
Hchl7s	50	77	428	**100**	134	18	26	19	25	36	44
okp3	33	49	**209**	113	122	29	38	35	>99	>99	>99
Hchl8s	17	35	253	68	**48**	73	45	44	>99	>99	>99
Hchl3s	46	57	39	**93**	130	11	21	85	80	82	87
Hchl2	25	77	45	**89**	144	5	8	18	49	50	65
CHL6	45	68	39	**91**	98	13	15	14	48	46	44
CHL7	23	78	**30**	105	116	8	8	5	26	27	44
Hchl6s	51	80	**36**	103	103	3	5	3	14	18	27
CHL1	32	64	**26**	115	120	14	16	14	68	71	72
CHL1s	32	64	21	**75**	121	14	13	6	62	39	61
okp5	11	12	12	**97**	99	1	2	2	19	21	21

H (%) – the percentage of all variables (i.e., cut and extraction) that were hybridised for C. H. and A. H.; *Mean Time (s/%)* – mean time spent to solve the instance, in seconds for N. H., and in a percentage relative to N. H. for both C. H. and A. H.; *CV* – coefficient of variation (also known as relative standard deviation) is the standard deviation for N. H., C. H., and A. H., divided by their respective means (CV is always a percentage); *Non-Root (%)* – the percentage of the total time which was *not* spent solving the root node. Source: the author.

The Clautiaux42 dataset has two traits that make it a worst-case scenario for both C. H. and A. H.: most instances are small instances, and the number of pieces sharing the same length, or width, is high. For the sake of comparison, let us define rr_l (rr_w) as the *repeat ratio* of the length (width) values of pieces. For a given set of piece types, the rr_l (rr_w) is a fraction with the difference between the set cardinality and the number of distinct length (width) values as the numerator, and the set cardinality minus one as the denominator. *Zero* means there is no repetition, and *one* means that all pieces share the same value in the respective dimension. The FMT59 dataset has $rr_l \approx 0.178$ and $rr_w \approx 0.155$, while the Clautiaux42 has $rr_l \approx 0.465$ and $rr_w \approx 0.402$. The metric is a good baseline, even if it does not account for some important details. For example, small piece types sharing a small length, or width, cause more hybridisation than large pieces sharing a large length (or width). This last observation is especially true for the BBA formulation, which has no cuts after the midplate.

Table 6.3 shows that C. H. hybridises only ≈5% of the variables and has minimal impact, while A. H. hybridises ≈55% of the variables but causes a large increase in both

time to solve and the number of cut variables. Table 6.3 reveals that for many instances, C. H. has less than 0.5% of hybridised variables, and the mean and CV show they behave basically the same as N. H. for most instances. In fact, the slight advantage of C. H. comes from the fact that the two hardest instances have no hybridisation and, therefore, the same mean time as N. H. But the third-hardest instance reaps a few seconds from the hybridisation of 9% of the variables. The extra variables from the A. H. have a strong negative effect on the mean time for all instances. For some instances (such as E05F18 and E00X23), the mean time is more than ten times longer than N. H. Finally, if rotation is allowed, then rr_l and rr_w become a single metric that can only be greater than or equal to both previous values; consequently, the gap in behaviour between C. H. and A. H. can only increase by allowing rotation.

Table 6.3 – Summary of hybridisation impact over BBA formulation and Clautiaux42 dataset.

Variant	**T. T.**	**$\triangle$ B. T.**	**#b**	**#extr.**	**#cuts**	**h %**	**k %**	**#plates**
N. H.	255.10	11.30	32	1,205	109,369	0.00	0.00	12,642
C. H.	251.58	7.78	10	1,205	109,369	5.50	2.11	12,642
A. H.	858.30	614.50	0	1,205	160,650	55.23	10.78	12,642

The description of the columns can be found in Table 6.1. Source: the author.

In general, for both datasets, C. H. either had a negligible difference from N. H. or provided some considerable benefit (especially for instances with longer running times). For solving mostly small instances, or instances with high rr_l or rr_w, the extra complexity brought to the formulation may not be worthwhile, but the change does not bring much risk of worsening the results. The A. H. has the best reduction of mean time and CV for both Hchl4s and Hchl8s (FMT59 dataset), but it has a consistently bad performance for small instances with high rr_l and rr_w. There is no clear class of instances for which it can consistently outperform C. H. (or N. H.).

Table 6.4 – Details of hybridisation impact over BBA formulation and Clautiaux42 dataset.

	H (%)		Mean Time (s/%)			CV (%)			Non-Root (%)		
Inst.	C	A	N (s)	C (%)	A (%)	N	C	A	N	C	A
E02F20	0	58	**83.70**	100	350	62	62	70	>99	>99	>99
E20F15	0	54	**68.26**	101	212	51	49	67	>99	>99	>99
E04F19	9	57	26.59	**64**	183	117	118	156	>99	>99	>99
E05F18	0	57	**8.52**	111	1,172	21	17	51	>99	>99	>99
E10X15	15	55	**7.62**	124	292	64	28	96	99	>99	>99
E08F15	0	51	**5.05**	115	424	34	33	63	99	99	>99
E04F17	0	54	**4.50**	100	730	16	16	7	99	99	>99
E02N20	0	55	**4.23**	100	721	17	17	45	99	99	>99
E05X15	9	51	3.46	**92**	213	25	25	31	99	99	99
E02F17	2	54	**3.43**	111	239	26	17	20	99	99	>99
E07F15	2	59	**3.27**	121	418	23	28	26	99	99	>99
E04F20	2	53	2.95	**76**	469	53	43	145	99	98	>99
E15N15	12	55	**2.78**	123	159	83	77	87	99	99	99
E07X15	0	51	2.55	**99**	300	17	18	45	99	98	99
E00X23	0	55	**2.26**	100	1,237	11	11	100	97	97	>99
E03X18	0	56	2.26	**86**	310	28	19	40	98	97	99
E04N17	9	49	**1.99**	128	234	40	35	31	98	98	99
E03N17	2	57	**1.98**	115	236	13	10	18	98	98	99
E04F15	1	49	**1.94**	101	182	14	9	13	98	98	99
E03N16	1	55	**1.78**	110	337	21	12	19	98	98	99
E02F22	4	54	**1.72**	101	321	41	38	55	98	98	99
E04N18	0	55	1.62	**86**	743	32	25	48	98	98	>99
E05F20	10	55	**1.36**	113	243	33	41	45	97	97	98
E00N23	0	49	1.33	**96**	215	12	15	22	96	96	98
E08N15	8	61	**1.19**	106	436	18	15	29	97	98	99
E05N17	0	53	**1.17**	112	364	18	15	52	97	97	99
E05F15	11	53	**1.10**	107	386	9	15	24	96	97	99
E15N10	13	53	1.09	**88**	236	20	30	24	97	97	98
E03N15	9	61	**1.09**	109	249	10	10	22	96	97	98
E05N15	1	55	**0.72**	113	200	33	15	21	96	95	97
E20X15	0	55	**0.71**	105	531	25	25	45	94	95	99
E07N15	8	63	**0.58**	116	313	44	46	105	97	97	98
E04N15	19	57	**0.48**	111	350	12	13	20	93	94	97
E13X15	0	57	0.48	**93**	205	19	18	19	94	91	96
E13N10	13	52	**0.37**	112	164	13	18	14	92	93	95
E00N15	1	53	**0.25**	104	134	8	8	11	76	77	83
E13N15	5	56	**0.24**	113	787	8	10	25	84	85	97
E07N10	32	59	**0.16**	119	158	37	22	13	87	89	88
E10N10	21	62	**0.14**	117	218	44	38	15	86	83	90
E10N15	29	52	**0.08**	117	148	11	9	8	66	67	73
E03N10	11	51	0.08	**95**	166	9	6	8	34	28	76
E00N10	31	61	**0.04**	132	199	4	2	109	15	22	61

The description of the columns can be found in Table 6.2. Source: the author.

7 RELATED PROBLEMS

The primary motivation for using a mathematical formulation as the solving method is *flexibility*. The possibility of adapting a formulation for other problems or variants often stays theoretical and occasionally is materialised and empirically examined. This chapter describes the changes necessary to adapt the proposed formulation (BBA) to the Guillotine 2D version of three other problems from the cutting and packing literature (the hybridisation from last chapter is not considered there). It also presents empirical results of these adaptations over suitable datasets and, in the last section, a proof of a generation mistake found in literature instances during preliminary experiments. As far as the author knows, the closest work to this thesis regarding this aspect is Bezerra et al. (2020) which adapts two MILP models from the literature to the 'two-dimensional level strip packing problem' (as they call it). Considering solving methods that are not formulations, there is also Fontan and Libralesso (2020) which adapts their anytime procedure for the two- and three-staged G2KP (exact and inexact variants) to the Guillotined 2D Bin Packing Problem and the Strip Packing Problem.

7.1 Problems definitions

The three related problems this chapter considers are the Guillotine 2D version of the Multiple Knapsack Problem (MKP), the Orthogonal Packing Problem (OPP), and the Cutting Stock Problem (CSP). The CSP is closely related to the Bin Packing Problem, BPP; the distinctions are discussed further ahead. Both the G2MKP and the G2CSP consider multiple original plates and, therefore, have homogeneous and heterogeneous variants. The G2MKP (or the G2CSP) is homogeneous if its multiple original plates share the same dimensions. It is heterogeneous if the original plates may come in distinct sizes (which may have a different cost associated with each). This work covers only the adaptation to the homogeneous variant, which is the simpler of the two options.

The (homogeneous) G2MKP is the generalization of the G2KP in which, instead of having a single copy of the original plate available, there are m copies of the original plate available. Nevertheless, both G2KP and the homogeneous G2MKP have only one original plate *type*, i.e., all the m original plate copies share the same dimensions (LxW). G2MKP instances only differ from G2KP instances by this extra parameter (m). Adaptation of G2KP instances to the G2MKP by an unmindful increase of m may lead to trivial

instances because it may become trivial to pack every piece.

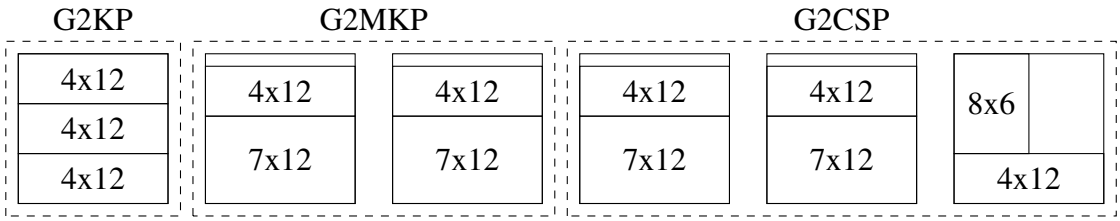
The G2OPP is the decision problem that inquires if a specific set of pieces can be packed into a plate of specific dimensions using only guillotine cuts. Compared to G2KP, the main differences of G2OPP instances are that (i) there are no piece profits, (ii) the piece demand is not an upper bound but a lower bound, and (iii) any instance with a summed piece area greater than the original plate area is trivially false and, therefore, not interesting. Piece profits may be dropped, and the demand may be reinterpreted, but the third distinction makes most instances of the G2KP uninteresting for the G2OPP.

The (homogeneous) G2CSP consists of minimizing the number of copies of the single original plate type while packing every copy of every piece type. The main differences between G2CSP and G2KP instances are that, in G2CSP instances: (i) there are no piece profits, (ii) the piece demand is not an upper bound but a lower bound, and (iii) trivial instances usually have the summed area of all pieces slightly larger than the closest multiple of the original plate area; such trait is not uncommon for G2KP instances. For example, if the area of the original plate type is 100, and the summed area of all pieces is 301, then there is a good chance of the optimal solution value to be 4 (i.e., the trivial area lower bound). The optimal solution cannot be less than four original plates. To need five or more original plates, the piece multiset would need to be designed to cause an uncommonly high waste ratio (i.e., greater than 25%). Moreover, there will probably exist not only one but many (nonsymmetrical) solutions able to pack all pieces into four original plates. This variety will make it easier for the solver to find an upper bound that matches the trivial lower bound.

Neither G2MKP nor G2CSP can be solved exactly by successively applying G2KP to the multiset of pieces yet unpacked. Figure 7.1 proves this statement and helps visualise the differences between these three problems. For the G2OPP, the example instance is trivially false.

The original one-dimensional Bin Packing Problem (BPP) and the one-dimensional CSP differ only in the expected diversity of the piece types in their respective instances. CSP instances typically have fewer piece types with a larger demand each, while BPP instances typically have more piece types with unitary (or very small) demand each. Often a method that can solve the CSP can also solve the BPP and vice-versa, albeit it may be more effective at a specific level of piece type diversity. The guillotine 2D variants of both problems are the same way, and, as such, the adapted formulation can solve both problems with low and high piece type diversity. A priori, the BBA formulation

Figure 7.1 – Visual proof G2KP cannot be iteratively used to solve G2MKP/G2CSP.



The original plate dimensions and the piece multiset are the same for the three problems: $L = W = 12$, $l = [4, 7, 8]$, $w = [12, 12, 6]$, $u = [3, 2, 1]$, and profits equivalent to their respective areas. For G2MKP, $m = 2$. G2CSP ignores the profits and consider u to be a lower bound. The only optimal solution for G2KP is a pattern that *cannot* pertain to any optimal solution of either of the other two problems. Souce: the author.

does not favour one problem variant over the other. The density of the cutting position discretization (and, therefore, size of the model) is not *directly* correlated to piece type diversity; it has more to do with the number of pieces with small dimensions and how many linear combinations of the piece dimensions give the same value. The G2CSP is also considered 'a generalization of the 2D-BPP in which equivalent items are grouped into demands' (<http://or.dei.unibo.it/library/2dpacklib>). If the definition above is adopted, our method solves the G2CSP, as it groups identical pieces into a single piece type with a demand value. For simplicity, the author chose to adopt G2CSP to refer to the problem to which the formulation was adapted. However, the experiments include instances of piece diversity on both sides of the spectrum, examining the impact of piece type diversity.

7.2 NP-hardness and NP-completenes of the related problems

This section discusses the NP-hardness and the NP-completeness of the related problems, and the G2KP is also mentioned. The homogeneous *G2MKP* and *G2CSP* are strongly NP-hard for similar reasons. The homogeneous one-dimensional MKP is the special case of the homogeneous G2MKP in which $W = w_1 = w_2 = \cdots = w_n$, i.e., the special case in which the second dimension is irrelevant. The analogue is valid for the homogeneous one-dimensional BPP and the G2BPP. (The G2CSP is equivalent to the G2BPP for the purpose of establishing NP-hardness.) The decision problem variant of the one-dimensional BPP is established as strongly NP-complete in Garey and Johnson (1979); consequently, the optimization version is strongly NP-hard, and its generalization, the G2BPP, is also strongly NP-hard. The one-dimensional MKP is strongly

NP-hard because it is a generalization of the Multiple Subset-Sum Problem with Identical Capacities, which is an optimization version of the strongly NP-complete 3-partition problem (MARTELLO; TOTH, 1990; KELLERER; PFERSCHY; PISINGER, 2010). There is a proof of the strong NP-completeness of the 3-partition problem in Garey and Johnson (1979) which is related to a previous proof from the same authors in Garey and Johnson (1975).

The 1D-BPP decision variant mentioned above is also a special case of the 'rectangle packing problem' (KORF, 2003). The G2OPP is the guillotined version of what was referred as the 'rectangle packing problem' in the past. The original proof holds for the G2OPP too because the guillotine constraint does not invalidate the proof. A summary of the proof follows. Given any instance of the 1D-BPP, the bin capacity B becomes W, and the upper bound on the number of bins necessary/available K becomes L, and each item becomes a piece of unitary length and width equal to the weight of the item. To solve this instance of the G2OPP is to solve the 1D-BPP variant mentioned. If, and only if, the G2OPP instance is unfeasible, the 1D-BPP variant is unfeasible. If the G2OPP instance is feasible, then the vertical position of the packed piece indicates the bin to which the piece was assigned, and the width of the original plate guarantees that the capacity of every bin (which is the same for all of them) is respected. As mentioned above, this 1D-BPP decision variant is strongly NP-complete, and therefore, G2OPP is also strongly NP-complete. The main problem studied in this work, the G2KP, is an optimization variant of the G2OPP and, therefore, is strongly NP-hard. The unconstrained G2KP is *not* strongly NP-hard because it allows for a piece to be obtained as many times as possible and, therefore, it cannot be used to solve the G2OPP. The unconstrained G2KP is a generalization of the Unbounded Knapsack Problem (UKP) which is also weakly NP-hard; the UKP is referred to as 'Integer Knapsack' in Garey and Johnson (1979, MP10).

7.3 BBA formulation adaptations

For the reader's convenience, the BBA formulation for the G2KP is replicated below[1] accompanied by a refresher on how it works and its notation. For consistency, the author chose to keep as much notation as possible. This choice means the input parameter u, which comes from *u*pper bound, was kept in the adaptation of the formulation to G2OPP and G2CSP, even if in these problems it indicates a *lower bound* instead.

[1]The formulation kept the same numbering from its original appearance.

The sets employed here are the same as before and keep their usual meaning: $\bar{J}$ – the set of piece types, $J \supseteq \bar{J}$ – the set of all plate types, $O = \{h, v\}$ – the set of cut orientations (horizontal and vertical), Q_{jo} – the sets of positions for which there is a cut of orientation o over plate j and, finally, E – the set of piece extractions. The following notation allow for easy access to pieces and plates related to the extractions: $E_{i*} = \{j : \exists\,(i,j) \in E\}$ (which plates may have a copy of i extracted from them) and $E_{*j} = \{i : \exists\,(i,j) \in E\}$ (which pieces may be extracted from a plate j).

$$\boldsymbol{max.} \sum_{(i,j)\in E} p_i e_{ij} \tag{3.9}$$

$$\boldsymbol{s.t.} \sum_{o\in O}\sum_{q\in Q_{jo}} x^o_{qj} + \sum_{i\in E_{*j}} e_{ij} \leq \sum_{k\in J}\sum_{o\in O}\sum_{q\in Q_{ko}} a^o_{qkj} x^o_{qk} \qquad \forall j \in J, j \neq 0, \tag{3.10}$$

$$\sum_{o\in O}\sum_{q\in Q_{0o}} x^o_{q0} + \sum_{i\in E_{*0}} e_{i0} \leq 1 \qquad , \tag{3.11}$$

$$\sum_{j\in E_{i*}} e_{ij} \leq u_i \qquad \forall i \in \bar{J}, \tag{3.12}$$

$$x^o_{qj} \in \mathbb{N}^0 \qquad \forall j \in J, o \in O, q \in Q_{jo}, \tag{3.13}$$

$$e_{ij} \in \mathbb{N}^0 \qquad \forall (i,j) \in E. \tag{3.14}$$

The domain of all variables is the non-negative integers (3.13)-(3.14). The value of a variable e_{ij} indicates the number of times a piece i was extracted from a plate j. An extraction only occurs if it respects the piece demand (3.12) (p_i is the profit of piece i) and, consequently, every extracted piece is taken into account by the objective function (3.9) which maximises the total profit (u_i is the demand of piece i).

The value of a variable x^o_{qj} indicates the number of times (distinct instances of) a plate j were cut at position q by a cut with orientation o. Both (3.11) and (3.10) handle which plates are available and, therefore, may be further cut or have pieces extracted from them. The only purpose of (3.11) is to make available one copy of the original plate (i.e., plate zero). For each other plate type j, (3.10) guarantees that, for each copy of j utilised for cutting or piece extraction, a copy of j was previously obtained from a larger plate. The number of plate j copies obtained by a cut at position q and orientation o over plate k is given by a^o_{qkj}, this listing is a byproduct of the plate enumeration.

Each of the following inner sections considers a different problem. For brevity, the adaptations do not replicate the necessary changes to allow piece rotation, but these

changes are the same as those described for G2KP in Section 3.5.

7.3.1 Adaptation to Multiple Knapsack Problem

To adapt the formulation for the G2KP to the (homogeneous) G2MKP, the only modification necessary is to replace the right-hand side of

$$\sum_{o \in O} \sum_{q \in Q_{0o}} x^o_{q0} + \sum_{i \in E_{*0}} e_{i0} \leq 1 \tag{3.11}$$

with the number of available original plates.

7.3.2 Adaptation to the homogeneous Cutting Stock Problem

To adapt the formulation for the G2KP to the homogeneous G2CSP, it is enough to introduce a new integer variable b and make the following changes to the formulation:

1. Replace the objective function (3.9) by $\boldsymbol{min.}\ b$.

2. Replace the literal 1 in the right-hand side of (3.11) by b.

3. reverse the sense of the demand constraint, i.e., eq. (3.12), from $\leq$ to $\geq$.

Formulations for CSP variants often need an extra constraint set to avoid a specific kind of solution symmetry. This class of symmetric solutions appears when the formulation numbers the bins and uses them as an index in some formulation variables. In such cases, each bin has a packing assigned to it; however, the choice of assigning packing a to bin 1 and packing b to bin 2 is, in fact, arbitrary: packing b could as well be assigned to bin 2 and packing a to bin 1. This is, two solutions may have the same number of bins and the same set of packings (piece multisets, or a tree of cuts) but, in the solution representation, the bins are *ordered* differently in relation to each other; even if such ordering is irrelevant for the problem. The FMT and BBA formulations do not manifest this issue and do not need an extra constraint set to fix it. In the FMT and BBA formulations, the bins are not represented by an index in any variables but by the value that flows into the variables from the right-hand side of (3.11). This representation does not allow for the discussed class of symmetric solutions: a solution with b bins and some specific set of

packings has only a single possible representation. The formulation also does not need to compute an upper bound on b, i.e., the number of bins necessary, but it makes it easy to provide one if available.

7.3.3 Adaptation to the Orthogonal Packing Problem

The employed adaptation of the formulation from the G2KP to the G2OPP consists only of two small changes. The first change consists in reversing the sense of the demand constraint, i.e., eq. (3.12), from $\leq$ to $\geq$, as done in the adaptation for the G2CSP. The second change consists in dropping the objective function. If the model is feasible, the solution to the decision problem is true; otherwise, it is false.

The adaptation for the G2OPP is not strictly necessary but an optimization. The G2KP formulation could be directly applied to suitable G2OPP instances. If absent, the piece type profit value may be set to be any positive value. If the optimal solution value for the G2KP is equal to $\sum_{i \in \bar{J}} u_i p_i$, then the answer to the decision problem (G2OPP) is true; otherwise, it is false. The issue with this approach is that the solver may waste time maximizing the profit after the upper bound has already lowered from $\sum_{i \in \bar{J}} u_i p_i$ but the lower bound did not yet match the upper bound. In this situation, the upper bound shows the solver already knows it is impossible to pack every piece (i.e., the answer to G2OPP is false), but this does not stop the solver; only proving which is the best profit value will stop it. This problem may be solved by passing a cutoff parameter to the solver or adding an extra constraint to the formulation (which forces the profit value to be equal to $\sum_{i \in \bar{J}} u_i p_i$). However, the author chose to change the demand constraint and drop the objective function for the experiments, as this adaptation better expresses the problem semantics to the solver.

7.4 Chosen datasets and experiment results

The author chose to use existing datasets from the literature when they were available and suitable to maximize the possibility of future direct or indirect comparisons. For the G2OPP and the G2CSP, suitable datasets were selected from the literature. The author did not find suitable datasets for the G2MKP in the literature and adapted previous instances from the literature to fill the gap. The computer setup and the Julia and Gurobi

versions/parameters are the same as described in Section 5.2, but the model was built and solved in the same process (i.e., there was no writing and reading from MPS file).

7.4.1 G2MKP

The G2MKP is the least studied of the three problems discussed in this chapter. As far as the author knows, the closest related work is Cui, Gu and Hu (2008) which proposes an exact algorithm for a special case of the three-staged G2MKP. The special case *disallow* the cuts of the third stage to generate two or more piece types from the same intermediary plate. The paper employs a subset of the FMT59 dataset adapted to have two or three original plates, i.e., $m \in \{2, 3\}$, most of them solved in less than half a second. The author of this thesis decided to not use these instances and, instead, adapted the A and CW datasets from the literature. A is an unweighted real-world dataset with high-demand piece types from Macedo, Alves and Carvalho (2010) which is also employed in the G2CSP experiments of this chapter. CW is a weighted artificial dataset with low-demand piece types from Fayard, Hifi and Zissimopoulos (1998) which is also employed in other G2KP experiments of this thesis. The adopted criteria for allowing an instance to have $m \in \{2, 4, 8\}$ is that the summed area of the original plates is at most half the summed piece area. This way, the *selection* aspect of the problem is retained.

Generally, for the adapted CW dataset, increasing the number of original plates available increases the time to solve, and allowing rotation (with the mirror plate enhancement) has a similar effect. These effects can be seen in Table 7.1. Increasing the number of original plates changes only a single right-hand side value, but allowing rotation causes a considerable increase in the model size, ranging from the ≈ 2.8 times of CW08_* to the ≈ 7.5 times of CW05_*. Both changes increase the search space, but this does not always translate directly to a larger gap between lower and upper bounds or a larger number of nodes needed to close it. In most cases, the time spent in the root node relaxation is relatively small, especially in the runs that end in timeout, as seen in Figure 7.5.

The time increase caused by rotation has a single exception, CW09_M2. For CW09_M2, the barrier times with and without rotation are, respectively, 45 and 3 seconds; the optimal solution, with and without rotation, are found, respectively, at 61 seconds (1.21% of gap) and 6 seconds (0.36% of gap); despite that, in 3864 nodes (206 seconds) the run allowing rotation closes the gap, while the run without rotation takes 317502 nodes (1525 seconds) to do the same. So, there is no other factor at play for this particular

Table 7.1 – Results for CW_M dataset (G2MKP) with and without rotation.

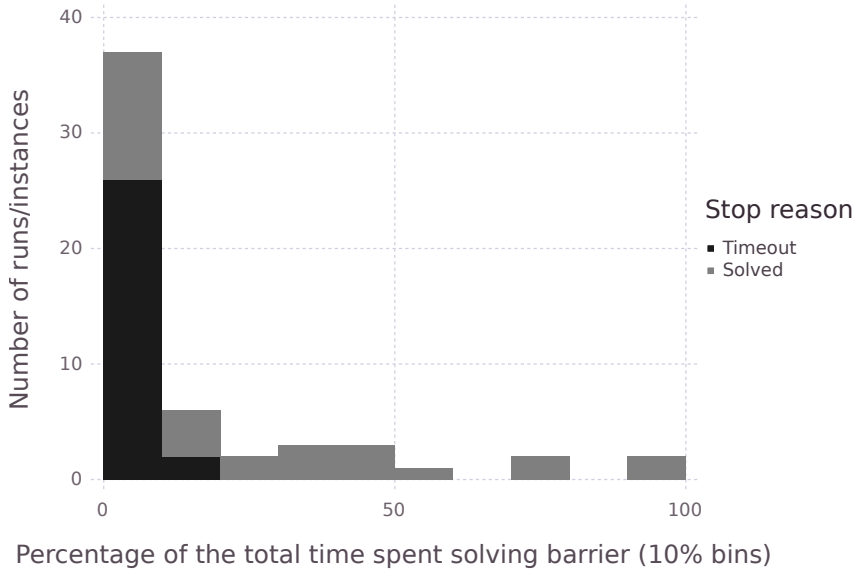
	No Rotation			R. & Mirror Plates		
Inst.	BKV	#nz	T. (s)	BKV	#nz	T. (s)
CW01_M2	12085	54,403	4.68	12227	150,308	254.11
CW01_M4	20093	"	>3600	20422	"	>3600
CW02_M2	10519	56,140	2.11	10943	143,098	16.81
CW02_M4	17384	"	>3600	17680	"	>3600
CW03_M2	10648	121,581	3.75	10877	491,790	69.33
CW03_M4	19113	"	10.38	19577	"	25.77
CW04_M2	11591	71,221	0.61	12412	497,740	23.66
CW04_M4	18104	"	57.29	19443	"	>3600
CW05_M2	21469	93,269	0.86	21657	700,304	835.05
CW05_M4	34358	"	43.38	35199	"	48.27
CW06_M2	23379	525,199	>3600	23639	2,872,208	>3600
CW06_M4	35407	"	>3600	35803	"	>3600
CW06_M8	52046	"	>3600	52674	"	>3600
CW07_M2	18834	123,851	3.28	19755	784,086	73.13
CW07_M4	33519	"	>3600	35007	"	>3600
CW07_M8	53631	"	>3600	54835	"	>3600
CW08_M2	8923	408,428	10.33	8985	1,174,315	>3600
CW08_M4	15074	"	684.15	15448	"	>3600
CW08_M8	24909	"	>3600	25241	"	>3600
CW09_M2	18712	204,212	1,525.83	19758	917,729	207.43
CW09_M4	30182	"	171.65	31174	"	>3600
CW09_M8	46695	"	>3600	47259	"	>3600
CW10_M2	12441	383,575	8.37	13047	2,714,389	98.02
CW10_M4	23065	"	>3600	24088	"	>3600
CW10_M8	37827	"	>3600	38655	"	>3600
CW11_M2	12078	384,237	7.16	12437	2,710,188	621.13
CW11_M4	22752	"	286.38	23419	"	>3600
CW11_M8	36691	"	96.72	37191	"	>3600

The best known value (BKV) is optimal if the run did not finish because timeout (indicated by *>3600*). *#nz* stands for number of nonzeros (which is always the same for instances adapted from the same original instance). The background color of the rows indicates the number of original plates.

instance except the branching needing fewer nodes to tighten the upper bound enough to close the gap.

The other set of exceptions (related to the change of m) has a behaviour similar to the case reported above. These exceptions are CW03_M2,4 (rotation), CW05_M2,4 (rotation), CW09_M2,4 (no rotation), and CW11_M4,8 (no rotation). In these cases, the time spent solving the root node is similar (respectively, 10s vs 9s, 17s vs 14s, 3s vs 2s, and 3s vs 3s), and the larger m takes the same or more time to find the optimal solution (respectively, 21s vs 22s, 27s vs 43s, 6s vs 15s, and 14s vs 49s). However, the percentage

Figure 7.2 – Impact of solving the root node in CW_M dataset (G2MKP).



The histogram has 10 bins, each bin aggregates the runs spending $(x, x + 10]\%$ of their time solving the root node ($x \in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]$). Souce: the author.

gap when the solver finds the optimal solution is either similar or lower (1.56 vs 0.35, 2.66 vs 0.23, 0.36 vs 0.33, and 0.58 vs 0.22 gap), which seems to result in lower total times (69s vs 26s, 830s vs 45s, 1525s vs 171s, 285s vs 95s). In most cases, the solver finds the optimal solution before starting branching (by employing heuristics). Therefore, the reported gap is against the root node relaxation (solved with any extra cuts added by the solver).

Considering only the instances for which both runs obtained an optimal solution (11 out of 28), the profit gap between optimal solution allowing and not allowing rotation ranges between 0.87% and 7.08% increase.

A final observation about the CW dataset is that CW10 and CW11 are very similar. They just have slightly different original plate dimensions (respectively, 992x970 vs 982x967), and their piece types have the same l, w, and p, i.e., just the demand u has different values. However, the timings of CW10 and CW11 are considerably distinct, showing how much depends on the specific values of an instance.

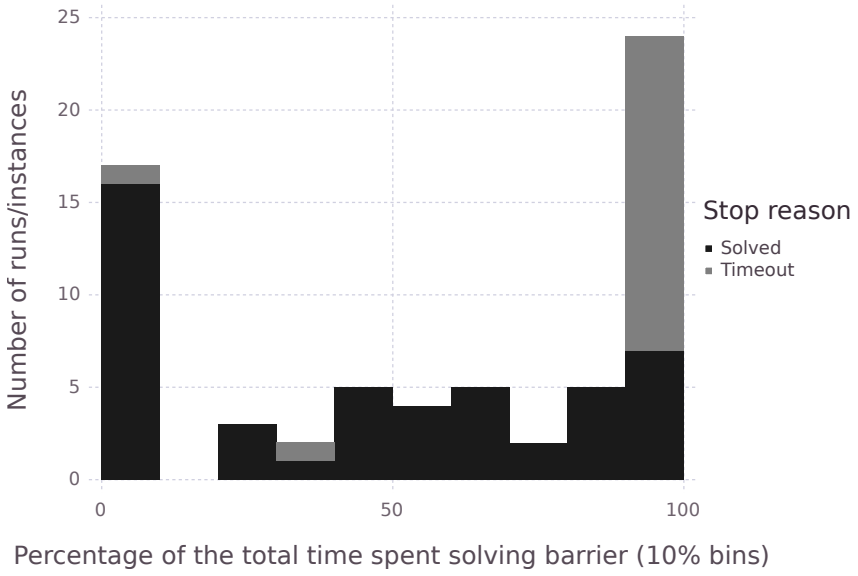
The author chose to exclude from Table 7.2 every instance for which both runs (with and without rotation) did not obtain a solution or obtained the trivial (i.e., empty) solution. Most of these runs failed due to memory exhaustion and did not even provide the number of nonzeros (i.e., failed before Gurobi loaded the model). Except by A21_M2, if an adapted instance was excluded for the reason above, then all the adapted instances

Table 7.2 – Results for A_M dataset (G2MKP) with and without rotation.

	No Rotation			R. & Mirror Plates		
Inst.	BKV	#nz	T. (s)	BKV	#nz	T. (s)
A03_M2	4,201,018	27	0.00	4,201,018	60	0.02
A17_M2	10,336,720	10,123	0.29	10,639,320	223,328	9.80
A26_M2	0	9,692,187	>3600	–	–	OOM
A31_M2	0	4,184,736	>3600	–	89,405,895	OOM
A34_M2	10,350,034	390,377	138.87	0	10,434,707	>3600
A36_M2	10,078,104	273,536	50.65	0	6,914,749	>3600
A37_M2	10,368,156	31,225	0.76	10,535,478	1,027,417	201.73
A42_M2	5,060,407	155	0.03	5,954,727	981	0.30
A43_M2	5,837,348	9,244	0.12	6,293,486	207,767	8.14
A05_M2	10,134,040	104,146	2.66	10,199,480	2,289,561	1,268.31
A05_M4	19,967,032	"	2.93	20,373,040	"	1,392.24
A07_M2	5,759,424	428	0.02	6,338,432	5,247	0.05
A07_M4	10,992,036	"	0.02	12,518,720	"	0.04
A12_M2	9,238,568	89	0.66	9,925,676	337	0.30
A12_M4	18,472,968	"	0.01	19,847,184	"	0.31
A13_M2	10,595,594	1,773,362	496.74	–	–	OOM
A13_M4	21,010,004	"	492.58	–	"	OOM
A23_M2	10,492,334	155,995	6.12	0	5,209,384	>3600
A23_M4	20,975,576	"	6.44	0	"	>3600
A40_M2	6,645,196	315,554	37.00	6,656,196	3,981,752	2,209.75
A40_M4	13,252,696	"	41.63	13,277,288	"	>3600
A41_M2	6,576,896	1,244,457	193.32	–	31,283,337	OOM
A41_M4	13,061,346	"	163.99	–	"	OOM
A02_M2	4,671,806	81	0.28	4,671,806	666	0.04
A02_M4	9,079,396	"	0.01	9,079,396	"	0.02
A02_M8	17,894,576	"	0.01	17,894,576	"	0.02
A15_M2	10,431,108	84,144	1.31	10,591,164	1,905,324	241.80
A15_M4	20,821,776	"	1.16	21,112,992	"	245.26
A15_M8	41,359,738	"	2.44	42,058,060	"	291.12
A20_M2	10,481,602	499,945	42.26	–	32,487,621	OOM
A20_M4	20,888,454	"	46.32	–	"	OOM
A20_M8	41,474,893	"	>3600	–	"	OOM
A21_M2	10,607,622	5,937,528	3,512.46	–	–	OOM
A21_M4	0	"	>3600	–	"	OOM
A21_M8	0	"	>3600	–	"	OOM

The columns are the same as of Table 7.1. OOM means *out of memory*. The instances are grouped by the number of adaptations, i.e., first, the instances that only have $m = 2$, then the ones with $m \in \{2, 4\}$; and then the ones with $m \in \{2, 4, 8\}$. Inside each group, the instances are sorted by name. Source: the author.

Figure 7.3 – Impact of solving the root node in A_M dataset (G2MKP).
(Only runs that did not suffer memory exhaustion.)



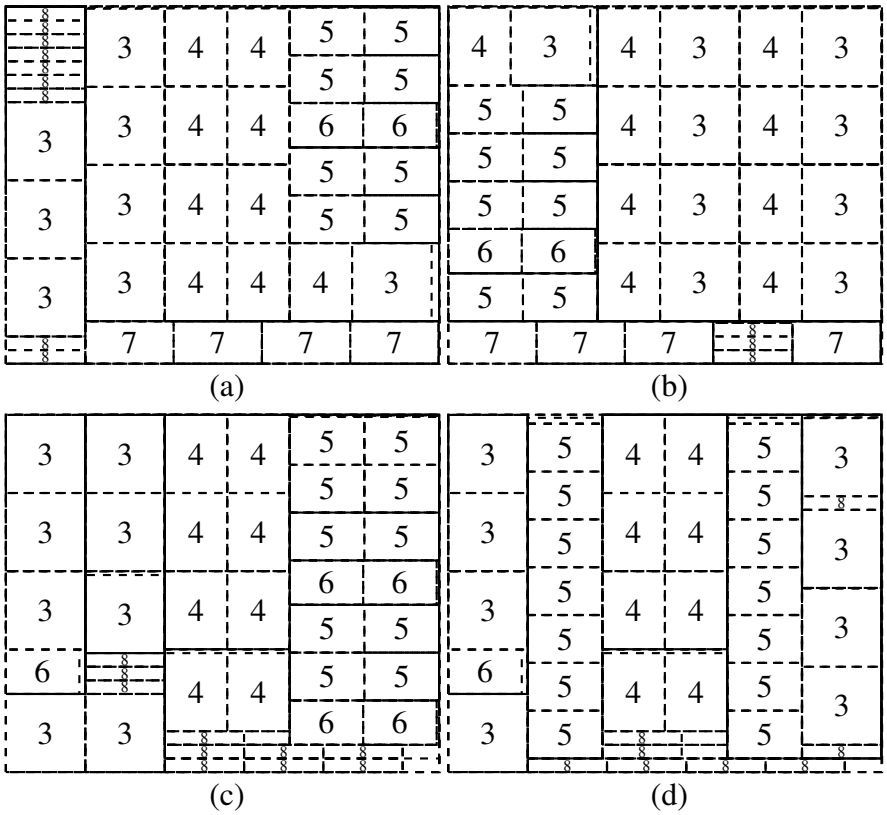
The histogram has 10 bins, each bin aggregates the runs spending $(x, x + 10]\%$ of their time solving the root node ($x \in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]$). Souce: the author.

sharing the same original instance (i.e., sharing the name before the underline) were also excluded for the same reason. This detail indicates that, in general, what determined if an instance was hard to solve were the other instance characteristics besides the m. The full list of adapted instances can be found in appendix A (under item *A-*_M2, A-*_M4, A-_M8*).

The A_M dataset behaves differently from the previous CW_M dataset. In the A_M runs, increasing m often has little effect on the solving times. Also, the time spent solving the root node is more significant, as shown in Figure 7.5. A more detailed examination of A13_M2,4 (no rotation) shows that almost all time is spent solving the root node (respectively, 457s and 419s); the optimal solutions are found shortly after (respectively, by the 484s and the 479s) with a very small gap (0.03% and 0.06%) which is soon closed.

One explanation for the similar timings, is that, given the instances are unweighted and have a high piece demand, the problem with $m = 2$ is not very distinct from the problem with $m = 4$. The optimal solutions found for A13_M2,4 corroborate this explanation: the original instance had $u = [10, 12, 120, 40, 80, 10, 8, 24]$, and the solutions for the $m \in \{2, 4\}$ instances employ the following numbers of pieces, respectively, $[0, 0, 16, 18, 16, 4, 8, 18]$ and $[0, 0, 34, 34, 40, 10, 8, 24]$, i.e., the demand for the last two piece types are the only ones low enough impede the $m = 4$ solutions to be just a repeti-

Figure 7.4 – Cutting patterns present in optimal solutions of A13_M{2,4} (G2MKP).



The numbers indicate the piece type index. The optimal solution of A13_M2 is two copies of pattern (a). The optimal solution of A13_M4 is two copies of pattern (b) plus a copy of (c) and another of (d). Souce: the author.

tion of the same pattern used in both plates of the $m = 2$ optimal solution. For reference, the patterns present in both solutions are presented in Figure 7.4.

The models from dataset A display a larger increase in the number of nonzeros if rotation is allowed (compared to the CW dataset). Another consequence of the large number of piece types with high demand. The smallest relative (and absolute) increase is of 2.22 times (from 27 to 60) from A03_M2; the largest relative increase is of 64.98 times (from ≈500,000 variables to ≈32,500,000) from A20_M{2,4,8}. The highest absolute increase does not figure in the table (from 4,184,736 to 89,405,895) because it pertains to instance A31_M2 runs, excluded for not obtaining any non-trivial solutions. Considering only the instances for which both runs obtained an optimal solution (18, all present in Table 7.2), it is possible to examine the profit gap between optimal solution allowing and not allowing rotation. Half (i.e., nine) of the instances presented an increase smaller than 1.7% (four of these had no increase); the largest increase was 17.67% by A42_M2, which is a small instance (eight piece types and 13 pieces total).

7.4.2 G2CSP

In this section, the following two datasets from the literature are employed. A is a real-world dataset with high-demand piece types from Macedo, Alves and Carvalho (2010) which the author of this thesis has also adapted for the G2MKP experiments (see the previous section). CLASS is an artificial dataset with low-demand piece types from both Berkey and Wang (1987) (first six classes) and Fayard, Hifi and Zissimopoulos (1998) (four last classes) which is not used in any other experiments of this thesis. The goal of choosing these two very distinct datasets is to test BBA over both G2CSP (represented by dataset A) and G2BPP (represented by dataset CLASS).

Table 7.3 presents a summary of other uses of dataset CLASS in the literature for the variant employed here and the non-exact three-staged variant. The author avoided usages of the dataset that did not include the guillotine constraint, solved the Strip Packing Problem, or focused entirely on heuristic results. The comparison is muddled by many factors: the exact problem variant, the machine, the solving method, the time limit, and how optimality is assessed. However, it seems clear that BBA (without any warm-start or given external bound) is not the most suitable solution method. The reasons for this inadequacy are made clear in the rest of the section, but they can be mostly summarised to: risk of memory exhaustion from the pseudo-polynomial model size; too much effort spent on an unnecessarily tight lower bound; closing the gap is mostly up to Gurobi internal primal heuristics. However, the BBA (with the mirror plate enhancement) had better results for the problem variant that allowed rotation than for the no-rotation variant.

The author only found two uses of dataset A in the literature that are relevant for the context of this thesis: Macedo, Alves and Carvalho (2010) (the dataset origin) and Mrad, Meftahi and Haouari (2013). In both works, the problem considered is the non-exact two-staged G2CSP without rotation, the method proposed is exact (respectively, MILP and branch-and-price), and the time limit is 7200 seconds. In Macedo, Alves and Carvalho (2010), their proposed MILP solves all instances within the time limit. The experiments of Mrad, Meftahi and Haouari (2013) show better performance of their branch-and-price compared to the aforementioned MILP; however, in their setup, the final results had both methods failing to solve all instances, i.e., worse results for the MILP than those reported in ist original wok.

The results found here cannot be directly compared to these previous results mainly because the search space of the non-exact two-staged variant is considerably smaller than

Table 7.3 – Known uses in the literature: dataset CLASS (G2CSP, no rotation)

Work	Problem	TL (s)	Machine	Method	LB	UB	#o
Alvelos et al. (2009)	non-exact three-staged G2CSP	7200	Intel Core 2 @ 2.13 GHz	Pseudo-polynomial MILP (CPLEX 11)	–	–	385
Fontan and Libralesso (2020)	non-exact three-staged G2CSP	60	Intel Core i5-8500 CPU @ 3.00GHz	Anytime Tree Search	–	7278	–
Pietrobuoni (2015)	G2CSP	1800	Intel Xeon E3-1220V2 @ 3.1 GHz	Heuristic (C)	7173	7281	393[a]
Pisinger and Sigurd (2007)	G2CSP	3600	Intel Pentium IV-3.0	Column Generation (Cplex 7)	7191	7266[b]	373–380[c]
This work	G2CSP	3600	AMD Ryzen 9 3900X @ 3.8 GHz	Pseudo-polynomial MILP (Gurobi 9.1)	–	–	344

[a] The method is heuristic and the optimality seem to be assessed by comparison of the obtained upper bound with a lower bound of external origin. [b] The exact source of lower bounds is not clear and can be a combination of distinct sources. [c] The number comes from an average multiplied by the number of groups. This work (as well some other works) do not have a LB or an UB value because the solving method failed for some instances without leaving either. Source: the author.

the one from the unlimited stages variant. The BBA solves only 24 instances (less than the other methods, even in the most disfavoring setup). Some failures are due to memory exhaustion, which does not deliver an LB or a UB. The MILP model from Macedo, Alves and Carvalho (2010) is also pseudo-polynomial, but memory exhaustion does not happen in their experiments because their problem variant is more manageable than ours. However, as the problem variant considered here allows for better packings, for some instances BBA found an optimal solution value that is better than the optimal solution value for the non-exact two-staged variant; these cases are: A-1 – 3 (4), A-7 – 13 (14), A-10 – 2 (3), A-13 – 13 (14), A-20 – 25 (27), A-23 – 13 (14), A-26 – 32 (35), A-39 – 3 (4), A-40 – 16 (17), and A-41 – 18 (19). The experiments demonstrate that BBA is more suitable for the G2BPP than the G2CSP.

The size of the models for dataset A increases by orders of magnitude when rotation is allowed. The exact amount is hard to estimate because this often leads to the model exhausting memory before it is fully loaded into Gurobi. Dataset A has a few

Table 7.4 – Results for dataset A (G2CSP) with and without rotation.

	No Rotation			R. & Mirror Plates		
Inst.	b	#nz	T. (s)	b	#nz	T. (s)
A-1	3	119	0.01	3	669	0.03
A-2	36	82	0.00	36	667	0.01
A-3	8	28	0.00	8	61	0.02
A-4	3	654	0.01	3	35,938	0.33
A-5	13	104,147	13.23	–	–	–
A-6	2	62	0.01	2	377	0.02
A-7	13	429	0.03	–	–	–
A-8	2	71	0.00	1	830	0.02
A-10	2	37,813	18.65	–	–	–
A-11	–	16,590,748	–	–	–	–
A-12	14	90	0.02	–	–	–
A-13	13	1,773,363	1,188.58	–	–	–
A-14	–	37,634,326	–	–	–	–
A-15	39	84,145	3.46	–	–	–
A-16	–	42,650,403	–	–	–	–
A-17	5	10,124	1.18	–	–	–
A-18	282	8,190,735	>3600	–	–	–
A-19	–	81,245,113	–	–	–	–
A-20	25	499,946	55.39	–	–	–
A-21	168	5,937,529	>3600	–	–	–
A-22	3	40	0.01	3	97	0.01
A-23	13	155,996	12.91	–	–	–
A-26	32	9,692,188	>3600	–	–	–
A-28	–	97,102,532	–	–	–	–
A-30	25	6,225,705	>3600	–	–	–
A-31	40	4,184,737	>3600	–	–	–
A-32	–	141,455,917	–	–	–	–
A-34	6	390,378	88.03	–	–	–
A-35	–	64,388,101	–	–	–	–
A-36	9	273,537	42.70	–	–	–
A-37	5	31,226	0.88	–	–	–
A-38	–	16,957,817	–	–	–	–
A-39	3	72,837	2.65	10	4,900,972	>3600
A-40	16	315,555	57.29	–	–	–
A-41	18	1,244,458	327.83	–	–	–
A-42	8	156	0.29	7	982	0.32
A-43	7	9,245	0.14	6	207,768	6.88

The instances in which both runs exhausted memory before the number of nonzeros was obtained are omitted. The description of the columns follows: b – value of variable b (i.e., the number of bins, the objective function) by the end of the optimization, which is optimal if the run time is smaller than the time limit (3600 seconds); #nz – number of nonzeros in the model; T. (s) – total time spent by the run in seconds. The dash indicates data disregarded because the run ended in memory exhaustion. Source: the author.

piece types, each with high demand, so often, a piece width did not exist as a piece length before rotation (and vice-versa). Consequently, allowing rotation may lead to doubling the number of distinct widths (lengths), each one of them associated with a piece with demand probably close to (or higher than) $\lfloor W/w_i \rfloor$ ($\lfloor L/l_i \rfloor$). The result is a much denser discretization and, therefore, a larger model.

The solver did not solve the no-rotation A31 model with 4,184,737 nonzeros within an hour, nor any model with more nonzeros. The solver suffered memory exhaustion in the no-rotation A11 model with 16,590,748 nonzeros and every model with more nonzeros. All runs that broke the time limit did not finish the root node relaxation and did not bring the lower bound up from zero; heuristics found the respective primal solutions during the solution of the root node.

Classes I to V present the opposite behaviour (compared to dataset A) when rotation is allowed: the number of nonzeros and the time to solve generally *decrease*. This behaviour can be explained by the small range from which the piece lengths and widths are sampled (respectively, $[1, 10]$, $[1, 10]$, $[1, 35]$, $[1, 35]$, and $[1, 100]$) and the small size of the original plate dimensions (squares of 10, 30, 40, 100, and again 100 units). With a small range of possible values, increasing the total number of pieces will quickly increase the number of values that appear both as a length and a width. Consequently, there is not a large difference between the discretization using only lengths and only widths (model without rotation) and the one with both sets combined for both dimensions (model with rotation). As the mirror plate enhancement can potentially reduce the number of nonzeros to one-fourth (after the increase caused by the denser discretization), the final result is often a model with rotation smaller than the respective model without rotation. This mechanism also explains why the exceptions to this behaviour appear only in the lower values of $\sum u_i$ (i.e., 20 and 40): if the length/width range is not yet saturated, combining lengths and widths increases the model size by more than the mirror plate can reduce it.

The shrinkage of the model size discussed in the last paragraph does not account alone for the decrease in the average time to solve. Allowing rotation may increase the number of distinct solutions sharing the optimal solution value or even allow for a better optimal solution (compared to the case without rotation). This behaviour helps close the gap because the initial lower bound often does not get looser; it often stays the same or goes down a single unit which matches the optimal value. One such example is the group of class V with 20 pieces: the model size increases by about 2.6 times, but the average time to solve decreases by about six times. Looking into the raw data, most of the

Table 7.5 – Results for CLASS I to VI (G2CSP) with and without rotation.

		No Rotation			R. & Mirror Plates		
C	$\sum u_i$	#opt	Avg. #nz	Avg. T (s)	#opt	Avg. #nz	Avg. T (s)
I	20	10	960	0.07	10	672	0.05
I	40	10	1,299	0.04	10	796	0.04
I	60	10	1,348	0.06	10	820	0.03
I	80	10	1,420	0.04	10	834	0.03
I	100	10	1,491	0.06	10	854	0.03
II	20	10	35,015	17.71	10	19,440	2.47
II	40	9	38,110	9.85	10	20,214	38.40
II	60	10	38,512	369.48	10	20,351	4.68
II	80	10	39,009	336.69	10	20,410	5.51
II	100	10	39,333	236.37	10	20,504	3.36
III	20	9	25,268	2.91	10	28,858	2.66
III	40	10	56,433	51.62	10	38,586	10.70
III	60	10	66,973	32.99	10	40,679	8.48
III	80	10	73,336	39.47	10	42,638	17.43
III	100	10	81,817	48.33	10	45,454	13.32
IV	20	7	1,028,785	1,836.98	9	640,680	1,164.56
IV	40	1	1,206,419	2,455.82	8	700,044	1,587.77
IV	60	0	–	–	3	727,716	1,989.13
IV	80	0	–	–	0	–	–
IV	100	0	–	–	3	734,838	2,179.76
V	20	9	95,331	115.80	10	252,312	18.43
V	40	10	448,653	513.98	10	457,204	176.21
V	60	7	596,699	313.21	9	474,356	158.92
V	80	8	779,755	857.55	10	566,326	582.90
V	100	4	949,798	1,016.19	10	630,620	540.01

The classes I to VI come from Berkey and Wang (1987). Class VI was suppressed because none of its associated runs was finished by optimality. Each line indicates ten runs. The *#opt* indicate how many runs found a solution and proved its optimality before the time limit. The average number of nonzeros (*Avg. #nz*) and the average time in seconds (*Avg. T (s)*) sum the respective characteristics of such runs and divide by *#opt*. The interrupted runs are not considered in the average because some were interrupted by timeout and others by memory exhaustion. Source: the author.

difference comes from instance `cl_05_020_05` which takes 27 seconds to solve with rotation and 885 without. The optimal solution of both runs is five bins, and both runs get the optimal lower bound from the root node relaxation, so it is just a matter of finding a matching primal solution.

All runs over instances of class VI failed to finish by optimality. The pieces employed in class VI are drawn from the same distribution as class V, but the original plate is nine times larger in terms of area (i.e., 100x100 vs 300x300). This increase in the original plate dimensions does not only mean a denser discretization and a larger search space, but

Table 7.6 – Results for CLASS VII to X (G2CSP) with and without rotation.

		No Rotation			R. & Mirror Plates		
C	$\sum u_i$	#opt	Avg. #nz	Avg. T (s)	#opt	Avg. #nz	Avg. T (s)
VII	20	10	31,155	96.22	10	98,189	10.33
VII	40	10	137,484	173.15	10	209,091	92.09
VII	60	10	198,334	597.51	10	277,052	231.71
VII	80	8	340,022	632.82	9	417,089	605.08
VII	100	4	540,266	1,316.47	7	465,485	861.68
VIII	20	10	14,089	5.46	10	99,028	7.50
VIII	40	10	113,393	77.37	10	205,848	51.61
VIII	60	9	197,823	328.11	10	280,897	397.35
VIII	80	7	353,320	440.39	8	429,091	440.19
VIII	100	6	427,341	987.87	9	460,246	516.39
IX	20	10	3,836	0.03	10	24,619	0.47
IX	40	10	47,227	1.33	10	105,356	3.61
IX	60	10	86,205	2.74	10	173,368	7.20
IX	80	10	269,340	19.69	10	362,968	19.87
IX	100	10	385,172	37.45	10	404,727	27.65
X	20	9	428,588	310.67	10	458,851	385.11
X	40	5	907,437	950.43	9	617,916	650.96
X	60	1	894,726	437.59	8	651,835	633.70
X	80	1	1,206,174	524.34	4	700,092	442.85
X	100	0	–	–	3	707,366	875.02

Classes VII to X come from Martello and Vigo (1998). See Table 7.5 for the description of the table layout. Source: the author.

it also means there are no more pieces equal to or larger than half a dimension (common in class V). These large pieces were easier to deal with than their smaller counterparts.

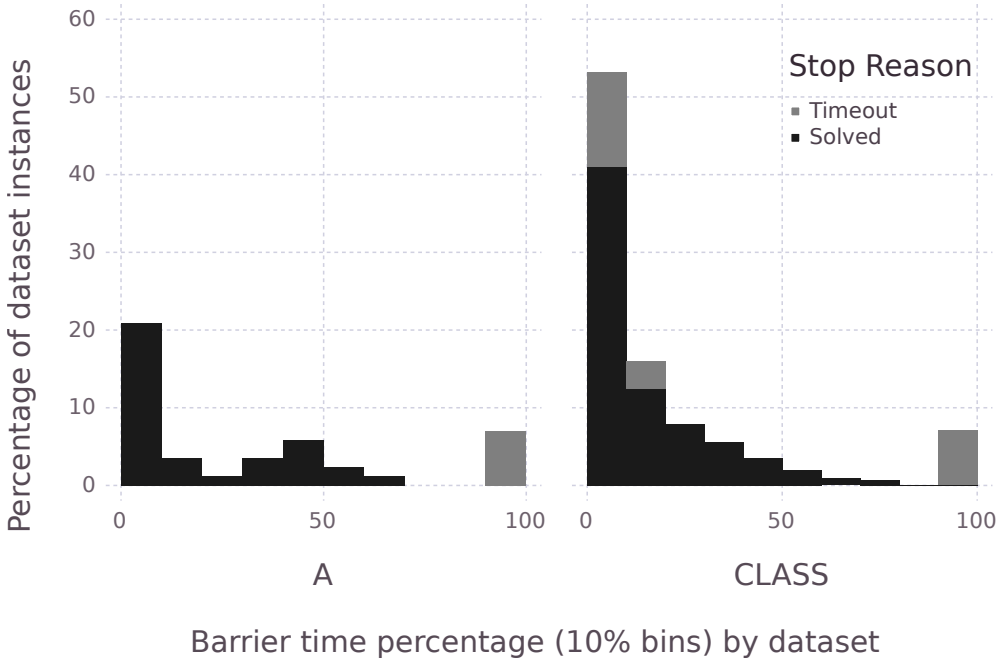
For classes VII to X, the effect of the mirror plate enhancement is less significant than in the previous classes: only class VII with 100 pieces and class X with more than 20 pieces show a reduction of the average model size when rotation is allowed. Nevertheless, there are many groups in which the models with rotation are larger on average, but the number of instances solved is higher and the average time to solve is lower than the models without rotation. The already mentioned advantage of models with rotation in obtaining good primal solutions explains such results. In fact, classes VII to VIII have length and width sampled from ranges with no intersection, which may hinder models without rotation, but it is not much relevant when the pieces are allowed rotation.

Classes VII to X share the same original plate dimensions (100x100); what changes is the distribution of the dimensions from which most of the pieces are sampled. Each instance of class IX has 70% or more of the pieces with both dimensions at *least* as large as half the original plate. Most of the solve time of such instances is spent at an easy but

large root node and with the solver unsuccessfully trying to improve the lower bound[2]. Each instance of class X has 70% or more of the pieces with both dimensions at *most* as large as half the original plate. The model can deal with such instances while the number of pieces is small; the problem is not the model size but finding a good primal solution (which is easier when rotation is allowed).

Considering only the instances of the whole CLASS dataset in which both the run with and without rotation finished by optimality, the total number of bins with and without rotation are, respectively, 5653 and 5846; this means rotation was able to reduce the number of bins needed by about 3.3%.

Figure 7.5 – Impact of solving the root node in A/CLASS dataset (G2CSP).



The histogram has 10 bins, each bin aggregates the runs spending $(x, x + 10]\%$ of their time solving the root node ($x \in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]$). The percentages do not sum up to 100% because runs killed by memory exhaustion count towards the total but are not assigned to any bin. Souce: the author.

All runs from both datasets that spent $90\text{~}100\%$ of total time in barrier finished because of the time limit. Considering the behaviour of the rest of the runs, if enough time was given for the run to finish, then these runs would probably not stay in the $90\text{~}100\%$ bin. This is, after finishing the barrier phase, the remainder of the run would probably take

[2]The author believes setting the Gurobi parameter `DegenMoves` to zero would reduce the time spent to solve class IX. However, the author deemed this too specific to have its own set of experiments.

about the same amount of time already spent in the barrier phase or more. All runs from both A and CLASS datasets that spent 90~100% of total time in barrier stopped because of the time limit. Considering the behaviour of the rest of the runs, if enough time for the run to the finish were given, then these runs would probably not stay in the 90~100% bin. After finishing the barrier phase, the remainder of the run would probably take even more time before proving optimality.

7.4.3 G2OPP

Two datasets are exclusively employed in this section: the CJCM from Clautiaux, Carlier and Moukrim (2007) and the dataset C (C1-p1 to C7-p3) from Hopper and Turton (2001). Both datasets are artificial and were generated for problems without the guillotine constraint, respectively, the Orthogonal Packing Problem (without rotation) and the Strip Packing Problem (with rotation). For the Orthogonal Packing Problem (OPP, i.e., G2OPP without the guillotine constraint), the feasibility status of each instance of both datasets is known: for CJCM, instances with 'F' in the name are feasible, instances with 'N' are not feasible, and instances with an 'X' have the same feasibility status than the instance of the row above them in Table 7.7; for C, all instances are feasible. Instances of CJCM also differ in their guaranteed percentage of waste, i.e., the relative difference between the summed piece area and the original plate area, which is indicated in the instance name by the first two digits after the letter 'E'. On the other hand, any feasible solution for the C instances needs a perfect fit, i.e., the summed piece area is the same as the original plate area. Finally, the CJCM instances are small and homogeneous (the total number of pieces is between 10 and 23, and all plates have 20x20 dimensions). In contrast, the number of pieces and plate dimensions of C instances increase with its category (the total number of pieces goes from 16 to 197 and original plate sizes from 20x20 to 160x240, see appendix for details).

Considering the OPP, the number of feasible instances in the CJCM is 14. Table 7.7 shows that with the guillotine restriction (but no rotation), this figure goes down to four, but then allowing rotation increases it to 40. To prove unfeasibility, the solver needs to exclude the whole search space from further consideration; on the other hand, to prove feasibility, the solver only needs to find a feasible solution. These particularities lead to the decrease in the average time to solve when rotation is allowed (without the mirror plate enhancement), even when this allowing rotation causes an increase in the

Table 7.7 – Results for dataset CJCM (G2OPP) with and without rotation.

	No Rotation			Rotation			R. +M.P.	
Inst.	F	#nz	T (s)	F	#nz	T (s)	#nz	T (s)
E00N10	N	3,330	0.32	N	7,040	0.39	3,759	0.44
E00N15	N	8,350	1.15	F	11,010	1.42	5,795	1.87
E00N23	N	9,052	2.33	F	11,316	3.99	5,949	1.16
E00X23	N	9,228	7.30	F	11,544	2.36	6,063	0.79
E02F17	N	8,197	9.26	F	10,820	3.79	5,699	3.42
E02F20	N	9,224	197.90	F	11,370	3.46	5,976	1.72
E02F22	F	8,768	3.44	F	11,396	9.18	5,989	1.20
E02N20	N	8,865	13.89	F	11,394	6.95	5,989	1.49
E03N10	N	5,729	0.10	N	9,862	3.62	5,211	0.86
E03N15	N	7,455	2.56	F	11,000	12.27	5,786	2.05
E03N16	N	7,072	5.88	F	9,842	5.11	5,204	3.26
E03N17	N	8,151	4.24	F	11,126	4.87	5,850	1.05
E03X18	N	9,057	4.53	F	11,410	13.06	5,998	2.04
E04F15	N	8,419	6.46	F	10,990	5.05	5,784	0.61
E04F17	N	9,050	10.89	F	11,218	3.17	5,901	0.28
E04F19	F	8,981	62.43	F	11,520	3.02	6,051	0.79
E04F20	F	9,090	10.73	F	11,256	2.68	5,919	1.32
E04N15	N	6,594	1.21	F	9,982	4.79	5,268	1.98
E04N17	N	8,847	8.50	F	11,378	2.54	5,979	1.93
E04N18	N	8,887	0.97	F	11,416	1.79	5,998	0.24
E05F15	N	8,571	4.94	F	11,412	1.68	5,996	0.13
E05F18	N	9,031	19.11	F	11,390	1.92	5,986	0.76
E05F20	F	9,080	5.85	F	11,454	2.53	6,018	0.31
E05N15	N	6,120	1.49	F	10,976	9.78	5,778	4.70
E05N17	N	8,885	2.57	F	11,328	31.84	5,955	0.96
E05X15	N	8,719	7.48	F	11,424	0.53	6,002	0.41
E07F15	N	6,657	5.70	F	10,896	3.88	5,734	1.01
E07N10	N	3,587	0.36	F	7,698	0.63	4,096	0.60
E07N15	N	4,947	1.73	F	10,310	1.74	5,432	2.63
E07X15	N	8,718	5.64	F	11,196	2.50	5,889	0.55
E08F15	N	8,281	12.65	F	10,886	2.26	5,734	0.18
E08N15	N	6,890	2.35	F	11,008	1.53	5,791	0.14
E10N10	N	3,732	0.34	F	9,346	1.03	4,950	0.33
E10N15	N	5,349	0.79	F	9,090	1.94	4,802	0.50
E10X15	N	7,490	43.90	F	10,602	2.10	5,583	0.28
E13N10	N	4,853	0.76	F	9,736	2.74	5,154	0.72
E13N15	N	8,496	0.42	F	11,356	1.36	5,968	0.45
E13X15	N	6,959	0.34	F	11,358	1.47	5,970	0.07
E15N10	N	5,563	2.21	F	10,458	1.57	5,512	0.85
E15N15	N	6,253	6.07	F	11,278	1.78	5,929	0.17
E20F15	N	8,923	62.95	F	11,126	0.83	5,852	0.11
E20X15	N	9,045	0.68	F	11,184	0.32	5,884	0.06
	4	7,583	12.91	40	10,771	4.04	5,671	1.05

F – feasible (F) or not feasible (N); #nz – number of nonzeros in the model; T. (s) – total time spent by the run in seconds. F is ommited in *R. +M.P.* (i.e., rotation with the mirror plate enhancement) because its is the same as F in *Rotation*. Source: the author.

Table 7.8 – Results for dataset C (G2OPP) with and without rotation.

	No Rotation			R. +Mirror Plates		
Inst.	F	#nz	T. (s)	F	#nz	T. (s)
c1-p1	F	5,787	0.40	F	4,031	0.68
c1-p2	N	9,728	0.72	F	5,419	0.95
c1-p3	F	6,879	0.12	F	3,985	0.44
c2-p1	F	19,154	10.95	F	20,755	2.33
c2-p2	N	22,332	267.73	F	20,495	16.50
c2-p3	F	17,860	9.47	F	20,863	9.81
c3-p1	F	87,001	235.73	F	81,168	139.34
c3-p2	N	109,977	2,865.11	F	95,354	1,742.43
c3-p3	F	102,175	29.41	F	95,355	140.72
c4-p1	X	273,229	>3600	X	141,522	>3600
c4-p2	X	307,507	>3600	F	159,145	2,938.40
c4-p3	X	291,911	>3600	X	159,475	>3600
c5-p1	X	561,197	>3600	X	438,745	>3600
c5-p2	X	584,310	>3600	X	439,306	>3600
c5-p3	X	584,032	>3600	X	435,995	>3600
c6-p1	X	1,259,079	>3600	X	976,512	>3600
c6-p2	X	1,403,995	>3600	X	1,044,021	>3600
c6-p3	X	1,297,746	>3600	X	1,034,700	>3600
c7-p1	X	10,908,645	>3600	X	8,083,056	>3600
c7-p2	X	11,376,814	>3600	X	8,389,704	>3600
c7-p3	X	11,154,288	>3600	X	8,362,393	>3600

Same columns as Table 7.7; 'X' indicates the run finished before proving either feasibility or unfeasibility. Source: the author.

model size. The halving of the average model size caused by enabling the mirror plate enhancement reduces the average time to solve even further.

The same behaviour observed in dataset CJCM can be observed in dataset C but to a smaller degree. Less dataset C instances are unfeasible without rotation (only three observed), and the mirror plate enhancement can only keep the models about the same size or at most about 25% lower. As of Category 4, the instances already have 49 pieces and 60x60 original plates, and only one instance is solved within the time limit. The generation method is the same for all categories, and the trend indicates that most instances are feasible (even more if rotation is enabled). Therefore, most instances from Category 4 and up fail because finding a single feasible solution is time-consuming.

Solving the root node relaxation (with the barrier method) takes more than 12.5% of the total time only in 12.5% of all the runs (i.e., merging both datasets) and, therefore, it is not a significant bottleneck.

Most papers that employ the two datasets considered in this section focus on a

method to solve the Strip Packing Problem, or the method is heuristic, or the method does not constrain the cuts to be guillotined, or any combination of these characteristics. This way, as far as the author knows, for contextualizing the results of the experiments, the most suitable works are Clautiaux, Jouglet and Moukrim (2011) *and* Fleszar (2016). Clautiaux, Jouglet and Moukrim (2011) proposes a new class of graphs called *guillotine graphs* which they use as the base for a constraint programming model that solves G2OPP instances through the ILOG solver. Fleszar (2016) propose a bottom-up pattern enumeration algorithm for the G2OPP, which takes advantage of dominance relations, enumeration order, and a heuristic look-ahead.

Unfortunately, while the two methods mentioned above are explicitly designed for the G2OPP, the instances of both datasets are solved as Strip Packing Problem instances by repeated solving them as G2OPP instances. The outer procedure consists of a loop that starts from either the upper or the lower bound (on the instance height), fixes the height, solves G2OPP as a subproblem, and either stops or iterates to the next height value. The difference between upper and lower bounds is often a few units, and in some cases (especially when rotation is allowed), the lower bound is the optimal solution value. Even so, a direct comparison with their result would be unfair to them. That said, the author will give a rough contextualization of results found here compared to theirs.

Clautiaux, Jouglet and Moukrim (2011) only solves the CJCM (without rotation) and names the instances differently from the names adopted in this work while missing instance E15N10 in the process. Fleszar (2016) reproduces the results from Clautiaux, Jouglet and Moukrim (2011), renaming the instances to the same standard used by us and will be used as reference. In Clautiaux, Jouglet and Moukrim (2011), three instances (E02F22, E04F19, and E04F20) match the lower bound adopted by them and, therefore, their method IGG_{lb} only solves G2OPP a single time. For these three instances, the timings of BBA and IGG_{lb} are, respectively, 3.44 vs 539.03, 10.73 vs 2.07, and 62.43 and 106.45. For the rest of the instances, there are many cases in which the IGG_{lb} solves for k distinct height values, and the time taken is $10k$ to $100k$ higher than the time taken by BBA to solve a single instance. There are hardly any cases in the opposite direction, i.e., BBA taking orders of magnitude more time than the mean time IGG_{lb} would take for a single G2OPP instance. However, run times can vary considerably even by changing one height unit, which makes such a comparison fragile. If anything, the evidence at least shows that the method to solve G2OPP proposed in Clautiaux, Jouglet and Moukrim (2011) does not dominate BBA.

The method of Fleszar (2016) takes at most 0.14 seconds to solve any instance in CJCM *for the Strip Packing Problem* and, consequently, dominates BBA in such dataset. For the C dataset, Fleszar (2016) solves the instances both with and without rotation (as done in this work too). When solving the instances without allowing rotation, their method hits the one-hour time limit in the same instances BBA also hits the time limit. However, when rotation is allowed, their method solves four more instances. When their method starts by the lower bound, which always matches the height adopted in the instances employed in this thesis, all runs that hit the time limit did not solve the G2OPP for the first height, i.e., the comparison between instances that hit the time limits is fair. For the C instances solved in both experiments, the timings of Fleszar (2016) are generally under a second and are better than ours. That said, as already mentioned, most instances that BBA failed to solve were also not solved in Fleszar (2016). One last observation is that Fleszar (2016) method has a worst-case memory requirement of $2(min\{W, L\}2^n)^2$ which is not directly comparable to BBA in which the worst-case the number of nonzeros is about $3/2(L + W) \times L \times W$ and the solver is free to explore the B&B tree different ways depending on the available memory. The size of the BBA model is not significantly affected by the number of piece types n, at least not directly, but it is significantly affected by the discretization on L and W and the number of unique values in l and w.

7.5 Other related problems

This section briefly discusses formulation adaptations for other related problem and variants. No implementation or experiments are provided for the additional adaptations discussed there.

7.5.1 Multiple Knapsack Problem (heterogeneous variant)

The adaptation of the BBA formulation from the G2KP to the heterogeneous G2MKP is not as straightforward as the adaptation to the homogeneous variant. First, the notation needs to be changed to account for multiple differently-sized original plates. In the previous notation, the only references to the original plate are to its length L, its width W, and to the fact it is the plate zero in J. For the sake of simplicity, let us assume plate-size normalization is enabled. Consider a set $K \subseteq J$ for representing the size-

normalized original plates and, for each $k \in K$, its (normalized) length L_k, (normalized) width W_k, and number of copies available U_k.

It is possible to avoid modifying the plate enumeration procedure by setting $L = max\{L_k : k \in K\}$, $W = max\{W_k : k \in K\}$, and plate zero to (L, W) (i.e., the enumeration is the same as before, but with the new dummy values). With the notation and plate enumeration procedure out of the way, the changes to the formulation boil down to replacing (3.11) by the following constraint set:

$$\sum_{o \in O} \sum_{q \in Q_{ko}} x^o_{qk} + \sum_{i \in E_{*k}} e_{ik} \leq U_k \qquad \forall k \in K \quad (7.1)$$

A specialized cut-and-plate enumeration procedure may have better performance but often the enumeration is not the performance bottleneck. Also, reduction procedures (or the solver itself) often will remove the unnecessary cuts and plates enumerated in this simplified fashion.

7.5.2 Strip Packing Problem

Differently from the other mentioned problems, the G2SPP does not define a W value a priori, as the problem searches the minimum W in which it is possible to pack all pieces. First, let us set W to a suitable upper bound, and then define our original plate (i.e., plate zero) as usual.

One straightforward adaptation, which does not directly alter plate enumeration, consists of the following steps: (i) add dummy pieces of width W and every normalized length, (ii) have the dummy pieces share the same one-unit demand (i.e., only one of these dummy pieces may be used in a solution), (iii) change the objective function to maximize the length of the selected dummy piece and, finally, (iv) change the demand constraint of all non-dummy pieces to be an equality. However, this adaptation is not ideal. For example, it introduces symmetries, as the dummy pieces, which simulate the unused width, may appear in the top, bottom, or middle of the pattern. Therefore, a better but not so straightforward adaptation, based on Furini, Malaguti and Thomopulos (2016), consists of the following changes:

1. L is set to be one unit greater than a suitable upper bound instead.

2. The original plate is not vertically discretized ($Q_{0v} = \emptyset$).

3. The original plate is horizontally discretized on its full extension.

4. The second child of every horizontal cut over the original plate is waste.

5. The demand constraint (3.12) becomes an equality, i.e., pieces are required.

6. Avoid direct extraction from the original plate, i.e., omit $\sum_{i \in E_{*0}} e_{i0}$ from (3.11).

7. The objective function changes from:

$$\boldsymbol{max.} \quad \sum_{(i,j) \in E} p_i e_{ij} \tag{3.9}$$

to:

$$\boldsymbol{min.} \quad \sum_{q \in Q_{0h}} q x^h_{q0} \tag{7.2}$$

In other words, the dummy original plate serves only to be horizontally cut (the chosen cut defines the objective function value), and the only child plate generated by the cut behaves as the original plate in the BBA formulation for the G2OPP.

7.5.3 The k-staged variant and the variant with defects

The author believes that adapting the BBA (or the FMT) formulations to either the variant with a limited number of stages, or the variants with defects in the original plate, is bound to be a not very fruitful endeavor. The reasoning behind this belief is that both variants prevent the formulation from disregarding the exact origin of each subplate and, therefore, break many reductions employed by the formulations.

In the case of the k-stage formulation, the formulation considers just one cut per plate at a time and do not group the set of cuts that could be done under a same guillotine stage together. It can be assumed that every horizontal (vertical) cut over a plate with width W (length L) can only pertain to the first stage; however, the children plates of the first cut of the second stage have lost all information that can help determine in which stage they were obtained. In fact, all plates of the same dimensions are conflated into a single number of plates available of some plate type, but each individual plate can have

been obtained at a different cutting stage. One of the strenghts of these formulations is exactly the model size reduction obtained by this conflation, which would need to be reconsidered to allow for an arbitrary k-stage limitation.

A variation of the same problem happens when plates with defects are considered. Now, it is not enough anymore for the plates to have the same size to be conflated into a single plate type, they need to have defects on the same exact spots. If there are few small defects, the model can work around this by generating extra plates types with defects and, once a subplate has no defects on it, from that subplate down the enumeration/model is the same as for the problem without defects. However, even if the defects are few and small, the plates with defects break some basic assumptions that allowed for many reductions in the enumeration. For an example, an horizontal cut at position q' of a plate of length l' cannot be assumed to yield the same results than a cut at position $l' - q'$ (i.e., its perfect symmetry) because the defects will be in different positions in each pair of child plates. This basic reduction is employed by both FMT and BBA formulations.

7.6 About the T instances from Hopper (2000)

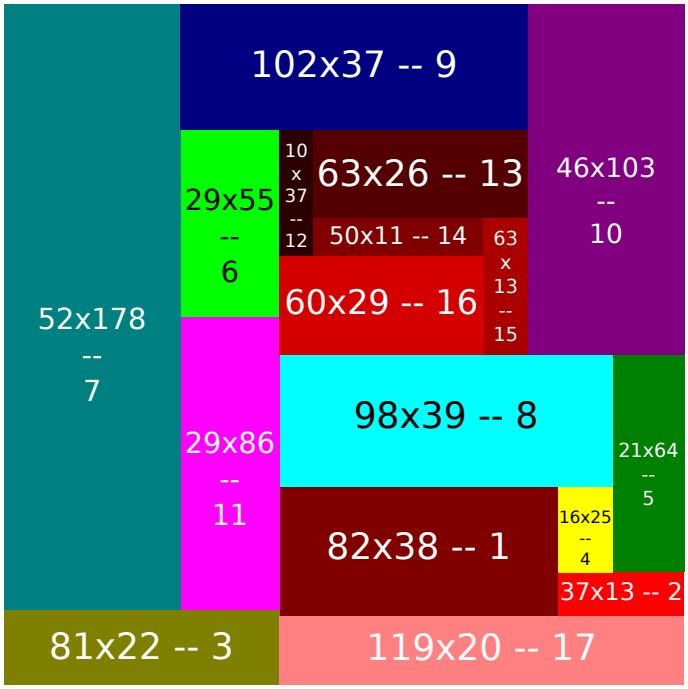
In a preliminary experiments phase, the author considered the instances of the datasets N and T from Hopper (2000). Each dataset has 35 instances and was generated by recursively cutting a 200x200 plate.

Hopper (2000) distinguishes between N and T solely by the cutting rules employed to generate each dataset. In N, a plate randomly selected to be further cut is divided into five subplates which follow the classic non-guillotinable pattern (see the pieces 12, 13, 14, 15, and 16 in Figure 7.6 for an example of such pattern). In T, single (vertical or horizontal) guillotine cuts should have been used instead. When the number of plates reaches the desired number, the process stops, and each plate is considered a piece for the generated instance. The process discards no plate; therefore, an optimal solution that packs every piece into a 200x200 plate must exist. However, only the process described for the T instances guarantees that optimal solutions can be obtained using only guillotine cuts.

Even with a three-hour time limit, only eight instances of dataset T (T1a, T1b, T1c, T1d, T1e, T2c, T2e, T3a) were solved optimally. However, in each of these eight runs, the solver returned an optimal pattern with less than all pieces of the respective instance. This unexpected result led us to manual analysis of the instances, which revealed

a mistake in their generation. The instances of the T dataset seem to have been generated using the same generator used for the N instances. Consequently, there is no guarantee of a wasteless solution using only guillotine cuts. In fact, for the first instance of the dataset T (T1a), the author can provide proof that it is impossible to obtain a guillotine packing of all pieces.

Figure 7.6 – A non-guillotinable optimal solution for instance T1a.



Souce: the author.

Proposition 7.1 *The entire piece set of the instance T1a cannot be packed into a space of 200x200 using only guillotine cuts.*

Proof: The desired pattern can only be wasteless, as the summed area of the pieces equals to the available area. Therefore, any guillotine cut that leads to some waste cannot lead to the desired pattern. Some piece (or piece subset) has to be positioned below and/or above piece 7 (52x178) for a pattern to have no waste. The only piece of length 22 is piece 3 (81x22), and no subset of pieces sums up length 22. As no other pieces sum up exactly length 22, the starting cut cannot be an horizontal cut separating pieces 7 and 3: the space by the sides of piece 3 would have some waste.

The only remaining option is a vertical cut separating pieces 7 and 3 from the rest of the plate. This cut must create a plate of width 81, so piece 3 can be obtained through a subsequent horizontal cut with no waste. Consequently, the desired pattern must have a subpattern of size 81x179 consisting of piece 7 and any other pieces (except piece 3) and with no waste. Such a

subpattern is impossible to obtain. The pieces able to fit by piece 7 sides are: 6 (29x55) and 11 (29x86), both with the exact width, and also the combination of pieces 4 (16x25) and 15 (13x40), that side-by-side sum width 29. Other pieces of small width (e.g., 5 and 12) cannot combine in a way that sum width 29. These pieces are insufficient to fill the gap. Therefore, there is no guillotinable pattern able to pack the entire piece set within the available area.

□

As far as the author knows, no previous paper has made this claim. Recently, Júnior et al. (2018) mentioned datasets N and T and their difference in generation; so the author believes this mistake is not widely known. The closest claim the author could find was from Wei et al. (2014) which says:

> "The set T consists of 70 instances whose known perfect packing follow guillotine-cut constraint. However, when we follow the method described by Hopper (2000) to restore the known perfect packing for T instances, we found the known perfect packing clearly does not follow the guillotine-cut constraint."

Unfortunately, this claim gets the number of instances wrong, so the author is not sure if they mixed T and N instances together or if they just committed a typo. Much of the prior work that solved instances from the T dataset focused either on heuristic methods or exact non-guillotine methods. Examples of such works areAlvarez-Valdes, Parreño and Tamarit (2008) (non-guillotine heuristic founds optimal solution for smaller T instances) and Wei et al. (2011) (non-guillotine heuristic). The author believes this explains why this discrepancy was not noticed sooner.

Some works may have been impacted by the mistake in the dataset generation. Bortfeldt and Jungmann (2012) proposes a tree-search algorithm that respects the guillotine constraint. The work compares optimality gaps of the proposed method and other non-guillotine methods. The work assumes all methods could reach the best values for such instances, which is not true. Consequently, the optimality gaps are wrong and unfair to the proposed method. Fortunately, the work does not use only the T dataset, so the problem is mitigated. Other works that appear to be in the same circumstances are Thomas and Chaudhari (2014) and Shang, Pan and Pan (2020).

Finally, even if the hypothesis about the generation mistake is correct, the author cannot claim that every instance in dataset T cannot have all their pieces extracted from a 200x200 plate using only guillotine cuts. The first iteration of the generation process divides the original plate into a guaranteedly non-guillotinable pattern. However, further cuts throw away the guarantee that the whole pattern cannot be rearranged in a guillotinable pattern. Intuitively, if the process continues for enough iterations, the pieces will be small enough to be rearranged into a guillotinable pattern without difficulty. For a trivial

example, any pattern with a non-guillotinable subpattern may be rearranged into a guillotinable pattern if (i) the length (or width) of the whole pattern is larger than the sum of the lengths (or widths) of each piece in the non-guillotinable subpattern and (ii) all the other pieces are of unitary size (1x1). Consequently, the generation process only makes it very likely but does not guarantee that the whole piece set cannot be packed back into the original plate with guillotine cuts alone.

8 CONCLUSIONS

The present work advances the state of the art on MILP formulations for the G2KP and related problems. This work proposes a (re-)formulation that improves the performance of one of the most competitive MILP formulations for the G2KP by at least one order of magnitude. The enhanced formulation dominates the original formulation in the instance set selected by the original formulation. Concerning other formulations for the problem in the literature, the proposed formulation has shorter run times, and it proves the optimality of more instances (in all datasets in which any of the considered formulations can prove the optimality of at least one instance). The weakness of the proposed formulation is that in harder datasets (like APT) it will either be unable to solve the root node under the time limit or fail by memory exhaustion. The other formulations cannot prove the optimality of instances of this dataset either, but they are able to return good solutions at least.

The proposed formulation and the plate-size normalisation are enhancements practically exempt from drawbacks except for the extra complexity of implementation, which is still lower than the pricing procedure they mostly supersede. The proposed hybridisation also does not add too much complexity to the implementation. However, it achieves moderate success only in runs where most time is spent after solving the root node, and its aggressive variant risks a large increase to run time for easier instances.

The flexibility advantage of formulations allows for easy adaptation for the rotation variant and changing the problem to G2MKP, G2OPP, and G2CSP. G2MKP is not deeply studied by the literature, and this thesis gives a starting point for future practitioners. For the G2CSP, as it is common in many CSP variants, a simple formulation has difficulty keeping with the state of the art. The relaxation upper bound is often optimal, but good upper and lower bounds methods can often prove optimality without needing a systematic framework for exploring the search space. The formulation can be a tool for scanning through the search space when these bounds cannot close the gap by themselves, but it is not competitive by itself. For the G2OPP, the proposed formulation seems competitive against a similar approach (a constraint programming model), but it is orders of magnitude slower than the state-of-the-art bottom-up pattern enumeration (which has the disadvantage of needing even more memory than the proposed formulation).

In the experiments, some elementary inferences were already discussed, such as the impact on the performance caused by the LP-solving algorithm, the specific changes

made, MIP-starting the models, some procedures proposed together with the original model (i.e., pricing and some preprocessing reductions), allowing rotation, and choice of the solver. The author deemed these too specific to be discussed here and better contextualised in their respective experiment sections.

The author believes symmetry-breaking plays a significant part in the success of the proposed formulation. In the experiments, the text focuses on the significant reduction of the model size because it is easier to measure. However, in Section 4.4, by comparing formulations with and without the *purge* procedure, it can be seen that a significant reduction of the model size does not always lead to a significant reduction in running times. In the case of the variables removed by the *purge* procedure (which could never assume a nonzero value), it seems clear the solver was able to disregard them without the need for the explicit removal by *purge*. The same does not apply to the variables removed by the enhanced model , the plate-size normalisation, the hybridisation, or even the (rotation-specific) mirror-plate enhancement), which could assume nonzero values and compose symmetric solutions. Each of these enhancements either removes redundant variables which could assume non-zero values or change the variables to further restrict the search space (without losing the guarantee of optimality). The enhanced formulation did not present consistent gains in the LP relaxation for them to be responsible for the observed improvement in performance. The author also believes the results suggest that clever dominance rules may considerably improve pseudo-polynomial models (which often have tight bounds but large formulations) before resorting to more complicated techniques (as the pricing procedure proposed in Furini, Malaguti and Thomopulos (2016), and described in Section 3.7, or column generation techniques).

The author believes it would be fruitful to pursue the following subjects in future research: applying the hybridisation to G2OPP (especially infeasible instances) as these are negatively impacted by the symmetries that hybridisation removes; any further reductions that may help to tackle large instances of the literature (as `gcut13`); and matheuristics that work without adaptation over any problems the formulation can solve and help to avoid the worst-case of the pseudo-polynomial formulations (i.e., providing no primal solution before the time limit if the root relaxation is hard to solve).

REFERENCES

ABED, F. et al. On Guillotine Cutting Sequences. In: . [s.n.], 2015. p. 1–19. ISBN 978-3-939897-89-7. Available from Internet: <https://kops.uni-konstanz.de/handle/123456789/33469>.

ADAMASZEK, A.; WIESE, A. A quasi-PTAS for the Two-Dimensional Geometric Knapsack Problem. In: **Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms**. Society for Industrial and Applied Mathematics, 2014, (Proceedings). p. 1491–1505. ISBN 978-1-61197-374-7. Available from Internet: <https://epubs.siam.org/doi/abs/10.1137/1.9781611973730.98>.

ALVAREZ-VALDÉS, R.; PARAJÓN, A.; TAMARIT, J. M. A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. **Computers & Operations Research**, v. 29, n. 7, p. 925–947, jun. 2002. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0305054800000952>.

ALVAREZ-VALDES, R.; PARREÑO, F.; TAMARIT, J. M. A branch and bound algorithm for the strip packing problem. **OR Spectrum**, v. 31, n. 2, p. 431–459, abr. 2009. ISSN 1436-6304. Available from Internet: <https://doi.org/10.1007/s00291-008-0128-5>.

ALVAREZ-VALDES, R.; PARREñO, F.; TAMARIT, J. M. Reactive GRASP for the strip-packing problem. **Computers & Operations Research**, v. 35, n. 4, p. 1065–1083, abr. 2008. ISSN 0305-0548. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S030505480600150X>.

ALVELOS, F. et al. Sequence based heuristics for two-dimensional bin packing problems. **Engineering Optimization**, v. 41, n. 8, p. 773–791, aug. 2009. ISSN 0305-215X. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/03052150902835960. Available from Internet: <https://doi.org/10.1080/03052150902835960>.

BANSAL, N. et al. A Structural Lemma in 2-Dimensional Packing, and Its Implications on Approximability. In: DONG, Y.; DU, D.-Z.; IBARRA, O. (Ed.). **Algorithms and Computation**. Berlin, Heidelberg: Springer, 2009. (Lecture Notes in Computer Science), p. 77–86. ISBN 978-3-642-10631-6.

BANSAL, N.; LODI, A.; SVIRIDENKO, M. A tale of two dimensional bin packing. In: **46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)**. [S.l.: s.n.], 2005. p. 657–666.

BEASLEY, J. E. Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. **Journal of the Operational Research Society**, v. 36, n. 4, p. 297–306, abr. 1985. ISSN 0160-5682. Available from Internet: <https://doi.org/10.1057/jors.1985.51>.

BEASLEY, J. E. An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure. **Operations Research**, v. 33, n. 1, p. 49–64, feb. 1985. ISSN 0030-364X. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.33.1.49>.

BELOV, G. **Problems, Models and Algorithms in One- and Two-Dimensional Cutting**. Thesis (PhD) — Fakultät Mathematik und Naturwissenschaften der Technischen Universität Dresden, sep. 2003. Available from Internet: <https://d-nb.info/970782489/>.

BELOV, G.; ROHLING, H. **A branch-and-price graph-theoretical algorithm for orthogonal-packing feasibility**. [S.l.], 2009. Available from Internet: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.156.7775&rep=rep1&type=pdf>.

BERKEY, J. O.; WANG, P. Y. Two-Dimensional Finite Bin-Packing Algorithms. **Journal of the Operational Research Society**, v. 38, n. 5, p. 423–429, may 1987. ISSN 1476-9360. Available from Internet: <https://doi.org/10.1057/jors.1987.70>.

BEZANSON, J. et al. Julia: A fresh approach to numerical computing. **SIAM Review**, v. 59, n. 1, p. 65–98, 2017. Available from Internet: <https://doi.org/10.1137/141000671>.

BEZERRA, V. M. R. et al. Models for the two-dimensional level strip packing problem – a review and a computational evaluation. **Journal of the Operational Research Society**, v. 71, n. 4, p. 606–627, abr. 2020. ISSN 0160-5682. Available from Internet: <https://doi.org/10.1080/01605682.2019.1578914>.

BORTFELDT, A.; JUNGMANN, S. A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint. **Annals of Operations Research**, v. 196, n. 1, p. 53–71, jul. 2012. ISSN 1572-9338. Available from Internet: <https://link.springer.com/article/10.1007/s10479-012-1084-7>.

BOSCHETTI, M. A.; MINGOZZI, A. The Two-Dimensional Finite Bin Packing Problem. Part II: New lower and upper bounds. **Quarterly Journal of the Belgian, French and Italian Operations Research Societies**, v. 1, n. 2, p. 135–147, jun. 2003. ISSN 1619-4500. Available from Internet: <https://doi.org/10.1007/s10288-002-0006-y>.

BOSCHETTI, M. A.; MINGOZZI, A.; HADJICONSTANTINOU, E. New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. **IMA Journal of Management Mathematics**, v. 13, n. 2, p. 95–119, abr. 2002. ISSN 1471-678X. Available from Internet: <https://academic.oup.com/imaman/article/13/2/95/686191>.

CHRISTENSEN, H. I. et al. Approximation and online algorithms for multidimensional bin packing: A survey. **Computer Science Review**, v. 24, p. 63–79, may 2017. ISSN 1574-0137. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1574013716301356>.

CHRISTOFIDES, N.; HADJICONSTANTINOU, E. An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. **European Journal of Operational Research**, v. 83, n. 1, p. 21–38, may 1995. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0377221793E02775>.

CHRISTOFIDES, N.; WHITLOCK, C. An Algorithm for Two-Dimensional Cutting Problems. **Operations Research**, v. 25, n. 1, p. 30–44, feb. 1977. ISSN 0030-364X. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.25.1.30>.

CLAUTIAUX, F.; CARLIER, J.; MOUKRIM, A. A new exact method for the two-dimensional orthogonal packing problem. **European Journal of Operational Research**, v. 183, n. 3, p. 1196–1211, dec. 2007. ISSN 0377-2217. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0377221706002992>.

CLAUTIAUX, F. et al. A new constraint programming approach for the orthogonal packing problem. **Computers & Operations Research**, v. 35, n. 3, p. 944–959, mar. 2008. ISSN 0305-0548. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0305054806001419>.

CLAUTIAUX, F.; JOUGLET, A.; MOUKRIM, A. A New Graph-Theoretical Model for the Guillotine-Cutting Problem. **INFORMS Journal on Computing**, v. 25, n. 1, p. 72–86, oct. 2011. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/ijoc.1110.0478>.

CLAUTIAUX, F. et al. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. **Discrete Optimization**, v. 29, p. 18–44, aug. 2018. ISSN 1572-5286. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1572528617301408>.

CLAUTIAUX, F. et al. Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. **EURO Journal on Computational Optimization**, v. 7, n. 3, p. 265–297, sep. 2019. ISSN 2192-4414. Available from Internet: <https://doi.org/10.1007/s13675-019-00113-9>.

CÔTÉ, J.-F.; IORI, M. The Meet-in-the-Middle Principle for Cutting and Packing Problems. **INFORMS Journal on Computing**, v. 30, n. 4, p. 646–661, nov. 2018. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/ijoc.2018.0806>.

CUI, Y.; GU, T.; HU, W. An algorithm for the constrained two-dimensional rectangular multiple identical large object placement problem. **Optimization Methods and Software**, v. 23, n. 3, p. 375–393, jun. 2008. ISSN 1055-6788. Available from Internet: <https://doi.org/10.1080/10556780701617163>.

CUNG, V.-D.; HIFI, M.; CUN, B. L. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. **International Transactions in Operational Research**, v. 7, n. 3, p. 185–210, 2000. ISSN 1475-3995. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2000.tb00194.x>.

DELORME, M.; IORI, M. Enhanced Pseudo-polynomial Formulations for Bin Packing and Cutting Stock Problems. **INFORMS Journal on Computing**, v. 32, n. 1, p. 101–119, jul. 2019. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/ijoc.2018.0880>.

DOLATABADI, M.; LODI, A.; MONACI, M. Exact algorithms for the two-dimensional guillotine knapsack. **Computers & Operations Research**, v. 39, n. 1, p. 48–53, jan. 2012. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0305054811000190>.

DUNNING, I.; HUCHETTE, J.; LUBIN, M. Jump: A modeling language for mathematical optimization. **SIAM Review**, v. 59, n. 2, p. 295–320, 2017.

DYCKHOFF, H. A New Linear Programming Approach to the Cutting Stock Problem. **Operations Research**, dec. 1981. Publisher: INFORMS. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.29.6.1092>.

FAYARD, D.; HIFI, M.; ZISSIMOPOULOS, V. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. **Journal of the Operational Research Society**, v. 49, n. 12, p. 1270–1277, dec. 1998. ISSN 1476-9360. Available from Internet: <https://doi.org/10.1057/palgrave.jors.2600638>.

FEKETE, S. P.; SCHEPERS, J. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: BURKARD, R.; WOEGINGER, G. (Ed.). **Algorithms — ESA '97**. Berlin, Heidelberg: Springer, 1997. (Lecture Notes in Computer Science), p. 144–156. ISBN 978-3-540-69536-3.

FLESZAR, K. An Exact Algorithm for the Two-Dimensional Stage-Unrestricted Guillotine Cutting/Packing Decision Problem. **INFORMS Journal on Computing**, v. 28, n. 4, p. 703–720, sep. 2016. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/10.1287/ijoc.2016.0708>.

FONTAN, F.; LIBRALESSO, L. An anytime tree search algorithm for two-dimensional two- and three-staged guillotine packing problems. 2020. Available from Internet: <https://hal.archives-ouvertes.fr/hal-02531031>.

FURINI, F.; MALAGUTI, E. A pseudo-polynomial size formulation for 2-stage 2-dimensional knapsack problems. In: **Proc. 45th Internat. Conf. Comput. Indust. Engrg.** [S.l.]: Curran Associates, Red Hook, NY, 2016. p. 833–840.

FURINI, F.; MALAGUTI, E.; THOMOPULOS, D. Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. **INFORMS Journal on Computing**, v. 28, n. 4, p. 736–751, oct. 2016. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/ijoc.2016.0710>.

G, Y.-G.; SEONG, Y.-J.; KANG, M.-K. A best-first branch and bound algorithm for unconstrained two-dimensional cutting problems. **Operations Research Letters**, v. 31, n. 4, p. 301–307, 2003. ISSN 0167-6377. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0167637703000026>.

GAREY, M. R.; JOHNSON, D. S. Complexity Results for Multiprocessor Scheduling under Resource Constraints. **SIAM Journal on Computing**, v. 4, n. 4, p. 397–411, dec. 1975. ISSN 0097-5397. Publisher: Society for Industrial and Applied Mathematics. Available from Internet: <https://epubs.siam.org/doi/abs/10.1137/0204035>.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. USA: W. H. Freeman & Co., 1979. ISBN 978-0-7167-1045-5.

GILMORE, P. C.; GOMORY, R. E. Multistage Cutting Stock Problems of Two and More Dimensions. **Operations Research**, v. 13, n. 1, p. 94–120, feb. 1965. ISSN 0030-364X. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.13.1.94>.

GRANDCOLAS, S.; PINTO, C. A sat encoding for multi-dimensional packing problems. In: LODI, A.; MILANO, M.; TOTH, P. (Ed.). **Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 141–146. ISBN 978-3-642-13520-0.

GáLVEZ, W. et al. Approximating Geometric Knapsack via L-Packings. In: **2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)**. [S.l.: s.n.], 2017. p. 260–271. ISSN: 0272-5428.

Herz, J. C. Recursive computational procedure for two-dimensional stock cutting. **IBM Journal of Research and Development**, v. 16, n. 5, p. 462–469, 1972.

HIFI, M. An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock. **Computers & Operations Research**, v. 24, n. 8, p. 727–736, aug. 1997. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0305054896000950>.

HIFI, M.; ROUCAIROL, C. Approximate and Exact Algorithms for Constrained (Un) Weighted Two-dimensional Two-staged Cutting Stock Problems. **Journal of Combinatorial Optimization**, v. 5, n. 4, p. 465–494, dec. 2001. ISSN 1573-2886. Available from Internet: <https://doi.org/10.1023/A:1011628809603>.

HOPPER, E. **Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods**. Thesis (PhD) — University of Wales, may 2000.

HOPPER, E.; TURTON, B. Problem generators for rectangular packing problems. **Stud. Inform. Univ.**, v. 2, n. 1, p. 123–136, 2002.

HOPPER, E.; TURTON, B. C. H. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. **European Journal of Operational Research**, v. 128, n. 1, p. 34–57, jan. 2001. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221799003574>.

IORI, M. et al. Exact Solution Techniques for Two-dimensional Cutting and Packing. **European Journal of Operational Research**, p. S0377221720306111, jul. 2020. ISSN 03772217. ArXiv: 2004.12619. Available from Internet: <http://arxiv.org/abs/2004.12619>.

JúNIOR, A. N. et al. The Two-Dimensional Strip Packing Problem: What Matters? In: VAZ, A. I. F. et al. (Ed.). **Operational Research**. Cham: Springer International Publishing, 2018. (Springer Proceedings in Mathematics & Statistics), p. 151–164. ISBN 978-3-319-71583-4.

KANG, M.; YOON, K. An improved best-first branch-and-bound algorithm for unconstrained two-dimensional cutting problems. **International Journal of Production Research**, v. 49, n. 15, p. 4437–4455, aug. 2011. ISSN 0020-7543. Available from Internet: <https://doi.org/10.1080/00207543.2010.493535>.

KELLERER, H.; PFERSCHY, U.; PISINGER, D. **Knapsack Problems**. Softcover reprint of hardcover 1st ed. 2004 edition. Berlin: Springer, 2010. ISBN 978-3-642-07311-3.

KORF, R. E. Optimal Rectangle Packing: Initial Results. In: **Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling**. Trento, Italy: AAAI Press, Menlo Park, California, 2003. v. 13, p. 287–295. ISBN 978-1-57735-187-0. Available from Internet: <https://www.aaai.org/Library/ICAPS/icaps03contents.php>.

LEUNG, J. Y.-T. et al. Packing squares into a square. **Journal of Parallel and Distributed Computing**, v. 10, n. 3, p. 271–275, nov. 1990. ISSN 0743-7315. Available from Internet: <http://www.sciencedirect.com/science/article/pii/074373159090019L>.

LODI, A.; MONACI, M. Integer linear programming models for 2-staged two-dimensional Knapsack problems. **Mathematical Programming**, v. 94, n. 2, p. 257–278, jan. 2003. ISSN 1436-4646. Available from Internet: <https://doi.org/10.1007/s10107-002-0319-9>.

MACEDO, R.; ALVES, C.; CARVALHO, J. M. Valério de. Arc-flow model for the two-dimensional guillotine cutting stock problem. **Computers & Operations Research**, v. 37, n. 6, p. 991–1001, jun. 2010. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0305054809002007>.

MARTELLO, S.; TOTH, P. **Knapsack Problems: algorithms and computer optimizations**. 1. ed. [S.l.]: John Wiley & Sons, Ltd., 1990. ISBN 0 471 92420 2.

MARTELLO, S.; VIGO, D. Exact Solution of the Two-Dimensional Finite Bin Packing Problem. **Management Science**, v. 44, n. 3, p. 388–399, mar. 1998. ISSN 0025-1909. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/mnsc.44.3.388>.

MARTIN, M. et al. Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. **International Transactions in Operational Research**, v. 27, n. 2, p. 767–793, 2020. ISSN 1475-3995. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12703>.

MARTIN, M. et al. Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. **International Transactions in Operational Research**, v. 27, n. 2, p. 767–793, 2020. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12703>.

MARTIN, M. et al. The constrained two-dimensional guillotine cutting problem with defects: an ILP formulation, a Benders decomposition and a CP-based algorithm. **International Journal of Production Research**, v. 0, n. 0, p. 1–18, jun. 2019. ISSN 0020-7543. Available from Internet: <https://doi.org/10.1080/00207543.2019.1630773>.

MARTIN, M.; MORABITO, R.; MUNARI, P. A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. **Computers & Operations Research**, v. 115, mar. 2020. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S030505481930293X>.

MARTIN, M.; MORABITO, R.; MUNARI, P. A top-down cutting approach for modeling the constrained two- and three-dimensional guillotine cutting problems. **Journal of the Operational Research Society**, v. 0, n. 0, p. 1–15, sep. 2020. ISSN 0160-5682. Available from Internet: <https://doi.org/10.1080/01605682.2020.1813640>.

MESSAOUD, S. B.; CHU, C.; ESPINOUSE, M.-L. Characterization and modelling of guillotine constraints. **European Journal of Operational Research**, v. 191, n. 1, p. 112–126, nov. 2008. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221707009083>.

MORABITO, R.; ARENALES, M. N. Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. **European Journal of Operational Research**, v. 94, n. 3, p. 548–560, nov. 1996. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/037722179500128X>.

MORABITO, R.; BELLUZZO, L. Optimising the cutting of wood fibre plates in the hardboard industry. **European Journal of Operational Research**, v. 183, n. 3, p. 1405–1420, dec. 2007. ISSN 0377-2217. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0377221706003134>.

MORABITO, R.; PUREZA, V. A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. **Annals of Operations Research**, v. 179, n. 1, p. 297–315, sep. 2010. ISSN 1572-9338. Available from Internet: <https://doi.org/10.1007/s10479-008-0457-4>.

MRAD, M.; MEFTAHI, I.; HAOUARI, M. A branch-and-price algorithm for the two-stage guillotine cutting stock problem. **Journal of the Operational Research Society**, v. 64, n. 5, p. 629–637, may 2013. ISSN 0160-5682. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jors.2012.70. Available from Internet: <https://doi.org/10.1057/jors.2012.70>.

NASCIMENTO, O. X. do; QUEIROZ, T. A. de; JUNQUEIRA, L. A MIP-CP based approach for two- and three-dimensional cutting problems with staged guillotine cuts. **Annals of Operations Research**, nov. 2019. ISSN 1572-9338. Available from Internet: <https://doi.org/10.1007/s10479-019-03466-x>.

OLIVEIRA, J. F.; FERREIRA, J. S. An improved version of Wang's algorithm for two-dimensional cutting problems. **European Journal of Operational Research**, v. 44, n. 2, p. 256–266, jan. 1990. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/037722179090361E>.

OPTIMIZATION, L. G. **Gurobi Optimizer Reference Manual**. 2020. Available from Internet: <http://www.gurobi.com>.

ORTMANN, F. G.; VUUREN, J. v. Modifed strip packing heuristics for the rectangular variable-sized bin packing problem. **ORiON**, v. 26, n. 1, 2010. ISSN 0529-191-X. Available from Internet: <https://www.ajol.info/index.php/orion/article/view/57310>.

PARREÑO, F.; ALONSO, M. T.; ALVAREZ-VALDES, R. Solving a large cutting problem in the glass manufacturing industry. **European Journal of Operational Research**, v. 287, n. 1, p. 378–388, nov. 2020. ISSN 0377-2217. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0377221720304446>.

PIETROBUONI, E. **Two-Dimensional Bin Packing Problem with Guillotine Restrictions**. Thesis (Tesi di dottorato) — alma, abr. 2015. Available from Internet: <http://amsdottorato.unibo.it/6810/>.

PISINGER, D.; SIGURD, M. Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem. **INFORMS Journal on Computing**, v. 19, n. 1, p. 36–51, feb. 2007. ISSN 1091-9856. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/ijoc.1060.0181>.

PUCHINGER, J.; RAIDL, G. R. Models and algorithms for three-stage two-dimensional bin packing. **European Journal of Operational Research**, v. 183, n. 3, p. 1304–1327, dec. 2007. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221706003067>.

RUSSO, M. et al. Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. **International Transactions in Operational Research**, v. 27, n. 2, p. 794–834, 2020. ISSN 1475-3995. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12687>.

RUSSO, M.; SFORZA, A.; STERLE, C. An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems. **Computers & Operations Research**, v. 50, p. 97–114, oct. 2014. ISSN 0305-0548. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0305054814000847>.

SCHEITHAUER, G.; TERNO, J. The g4-heuristic for the pallet loading problem. **Journal of the Operational Research Society**, Taylor & Francis, v. 47, n. 4, p. 511–522, 1996. Available from Internet: <https://doi.org/10.1057/jors.1996.57>.

SHANG, Z.; PAN, M.; PAN, J. An improved priority heuristic for the fixed guillotine rectangular packing problem. v. 1656, p. 012005, sep. 2020. ISSN 1742-6596. Publisher: IOP Publishing. Available from Internet: <https://doi.org/10.1088/1742-6596/1656/1/012005>.

SILVA, E.; ALVELOS, F.; CARVALHO, J. M. Valério de. An integer programming model for two- and three-stage two-dimensional cutting stock problems. **European Journal of Operational Research**, v. 205, n. 3, p. 699–708, sep. 2010. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221710000731>.

SONG, X. et al. A worst case analysis of a dynamic programming-based heuristic algorithm for 2D unconstrained guillotine cutting. **European Journal of Operational Research**, v. 202, n. 2, p. 368–378, abr. 2010. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221709003944>.

TERNO, J.; LINDEMANN, R.; SCHEITHAUER, G. **Zuschnittprobleme und ihre praktische Lösung**. Thun und Frankfurt/Main: Verlag Harri Deutsch, 1987.

THOMAS, J.; CHAUDHARI, N. S. Design of efficient packing system using genetic algorithm based on hyper heuristic approach. **Advances in Engineering Software**, v. 73, p. 45–52, jul. 2014. ISSN 0965-9978. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0965997814000489>.

THOMOPULOS, D. **Models and Solutions of Resource Allocation Problems based on Integer Linear and Nonlinear Programming**. Thesis (PhD) — University of Bologna, may 2016. Available from Internet: <http://amsdottorato.unibo.it/7399/>.

TSCHÖKE, S.; HOLTHÖFER, N. A new parallel approach to the constrained two-dimensional cutting stock problem. In: FERREIRA, A.; ROLIM, J. (Ed.). **Parallel Algorithms for Irregularly Structured Problems**. Berlin, Heidelberg: Springer, 1995. (Lecture Notes in Computer Science), p. 285–300. ISBN 978-3-540-44915-7.

VELASCO, A. S.; UCHOA, E. Improved state space relaxation for constrained two-dimensional guillotine cutting problems. **European Journal of Operational Research**, v. 272, n. 1, p. 106–120, jan. 2019. ISSN 0377-2217. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0377221718305393>.

VISWANATHAN, K. V.; BAGCHI, A. Best-First Search Methods for Constrained Two-Dimensional Cutting Stock Problems. **Operations Research**, v. 41, n. 4, p. 768–776, aug. 1993. ISSN 0030-364X. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.41.4.768>.

WANG, P. Y. Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. **Operations Research**, jun. 1983. Available from Internet: <https://pubsonline.informs.org/doi/abs/10.1287/opre.31.3.573>.

WEI, L. et al. A skyline heuristic for the 2D rectangular packing and strip packing problems. **European Journal of Operational Research**, v. 215, n. 2, p. 337–346, dec. 2011. ISSN 0377-2217. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0377221711005510>.

WEI, L. et al. A block-based layer building approach for the 2D guillotine strip packing problem. **European Journal of Operational Research**, v. 239, n. 1, p. 58–69, nov. 2014. ISSN 0377-2217. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0377221714003464>.

YANASSE, H. H.; MORABITO, R. A note on linear models for two-group and three-group two-dimensional guillotine cutting problems. **International Journal of Production Research**, v. 46, n. 21, p. 6189–6206, nov. 2008. ISSN 0020-7543. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207540601011543. Available from Internet: <https://doi.org/10.1080/00207540601011543>.

YOON, K.; AHN, S.; KANG, M. An improved best-first branch-and-bound algorithm for constrained two-dimensional guillotine cutting problems. **International Journal of Production Research**, v. 51, n. 6, p. 1680–1693, mar. 2013. ISSN 0020-7543. Available from Internet: <https://doi.org/10.1080/00207543.2012.693965>.

APPENDIX A — DETAILS ON MENTIONED DATASETS

The following is a list of every dataset that provided at least one of the instances used in this thesis experiments, as well as the N and T datasets, discussed in Section 7.6. The list is sorted by the year of the work that proposed the first (if not all) instances of the dataset.

HH *Proposed in:* Herz (1972) and Hifi (1997) (see details below) *for the* unweighted unconstrained (and, after, constrained) G2KP, and the *first known reference to the name adopted in this work is* Cung, Hifi and Cun (2000). *Other names:* the proposing paper does not name the unconstrained version and Hifi (1997) call the constrained version of H. *Characteristics:* "[...], we have considered another instance (denoted H), derived from the instance of Herz (1972), by adding an upper bound for each piece. The instance is described by $(L, W) = (127, 98)$, $n = 5$, $b = (5, 4, 2, 1, 6)$, $c_i = l_i \times w_i$ and (l_i, w_i), for $i = 1 \ldots 5$, are given by $(21, 13)$, $(36, 17)$, $(54, 20)$, $(24, 27)$ and $(18, 65)$, respectively." (HIFI, 1997). Their b is referred to as u in this thesis (i.e., piece demand), the analogue is valid for c and p (i.e., piece profit).

cgcut1 to cgcut3 *Proposed in* Christofides and Whitlock (1977) *for the* G2KP, and the *first known reference to the name adopted in this work here is* Martello and Vigo (1998). *Other names:* the instances were numbered as 1–3 by the proposing paper, which is a common approach but often not considered a name, in this case, however, other papers (e.g., Hifi (1997)) and instance repositories (e.g., <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting>) adopted the numbers as names; also, Fayard, Hifi and Zissimopoulos (1998) refers to cgcut1–2 as CHW1–2, Tschöke and Holthöfer (1995) proposes a four-instance unnamed dataset in which instances 1 and 3 are cgcut1 and cgcut3, consequently, cgcut1 and cgcut3 are also called STS1 and STS3 by PackLib^2 (<https://www.ibr.cs.tu-bs.de/alg/packlib/xml/b-autdg-85-xml.shtml>); Velasco and Uchoa (2019) mention a CW4 instance from Christofides and Whitlock (1977), however, the author believes they wanted to refer to the CW4 instance from Fayard, Hifi and Zissimopoulos (1998) instead. *Characteristics:* The cgcut1–3 are part of a larger semi-artificially generated dataset, but they were the only ones fully described in the body of the original paper and the most commonly adopted by later works. The authors selected the number of pieces (7, 10, and 20, respectively) and the original plate dimensions (15x10, 40x70, and

40x70, respectively). The piece dimensions were obtained by defining selecting a random value in $[1, 0.25 \times L \times W]$ to be the piece area, then selecting a random integer value between one and the piece area to be the piece length and, finally, determined the piece width in base of the already defined piece length and the provisional area (rounding the width up, if necessary, i.e., allowing the area to grow instead of shrink). The profit values were obtained by multiplying the area by a random real number between 1 and 3. Finally, the demand vector was handpicked by the authors to best suit their purposes.

wang20 *Proposed in:* Wang (1983) *for the* unweighted G2KP, i.e., waste minization variant, the paper also cover the G2CSP but the instance does not seem to be used for this purpose, and the *first known reference to the name adopted in this work is* Fekete and Schepers (1997). *Other names:* the instance is also referred as W by Fayard, Hifi and Zissimopoulos (1998), however, in <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>, W has a different demand vector (which does not seem to affect the optimal objective value). *Characteristics:* The instance is, according to the author, "a variation of an example presented by Christofides and Whitlock (1977)". The instance is fully described in the proposing paper and reproduced here: $L = 70$, $W = 40$, $l = [11, 12, 14, 17, 18, 21, 23, 24, 24, 25, 27, 32, 34, 35, 36, 37, 38, 39, 41, 43]$, $w = [19, 21, 23, 9, 29, 31, 33, 15, 15, 16, 17, 22, 24, 25, 26, 27, 28, 29, 30, 31]$, $u = [4, 3, 4, 1, 3, 3, 3, 1, 2, 4, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1]$.

gcut1 to gcut13 *Proposed in* Beasley (1985a) *for the* unconstrained G2KP with both a limited and an unlimited number of stages, and the *first known reference to the name adopted in this work here is* Martello and Vigo (1998). *Other names:* gcut13 is also referred to as B by Fekete and Schepers (1997). *Characteristics:* The profit of each piece is set to their area, as in the unweighted variant, and the demand of each piece was left undefined, as they were developed for an unconstrained variant. In their experiments, Furini, Malaguti and Thomopulos (2016) set the demand of each piece to one, and the same is done here (i.e., in the experiments of this work), to allow the comparison with their results. The gcut1–12 instances are artificially generated: the number of sampled pieces is $[10, 20, 30, 50]$ for the first four, middle four, and last four instances; $L = W$ and they are 250 for the first four, 500 for the middle four, and 1000 for the last four; both piece length and width are sampled from an integer uniform distribution $[L/4, 3L/4]$. The gcut13 instance is a real-world instance of

32 pieces and an original plate of size 3000x3000, fully described in the Table 2 of the original paper. This assimetry is probably the cause many papers select only the gcut1–12 for their experiments. *Online repositories:* PackLib[2] <https://www.ibr.cs.tu-bs.de/alg/packlib/xml/b-autdg-85-xml.shtml>, and ESICUP <https://www.euro-online.org/websites/esicup/data-sets/>.

cl_* (a.k.a. CLASS) *Proposed in:* Berkey and Wang (1987) (first 6 classes with 50 instances each) and Martello and Vigo (1998) (last 4 classes with 50 instances each) *for the* two-dimensional BPP (non-guillotine) and the *first known reference to the name adopted in this work is* hard to pinpoint, Martello and Vigo (1998) refer to each new set of 50 instances they propose as classes, and Boschetti and Mingozzi (2003) echoes this when they merge some instance sets from the two previous works together into a single dataset divided into 10 classes, but the name *CLASS* (also adopted by 2DPackLib) seems clearly accidental. *Other names:* in (ALVELOS et al., 2009), the initials of the authors are used for each part of the dataset (BW for the first six classes, and MV for the four last classes), but other names are also possible, as the instance groups were just numbered (using roman numerals) and the individual instances are referred just by their attributes (i.e., cl_*class_n_seed*, where n is the total number of pieces). *Characteristics:* each class has 50 instances, these instances can be further divided into five groups, each group has 10 instances sharing the same $n \in \{20, 40, 60, 80, 100\}$, all instances inside a group have exactly the same generation parameters except by the random seed. For the first six classes, both the length and the width of the pieces is sampled from an integer uniform distribution $[1, 10]$ (classes I and II), $[1, 35]$ (classes III and IV), and $[1, 100]$ (classes V and VI), and the original plates are squares of, respectively, 10, 30, 40, 100, 100 (again), and 300 units; for the last 4 classes all original plates are 100x100, and the piece dimensions are drawn from four *types* of piece size distribution. Each piece from the class (originally referred to as) $k \in \{I, II, III, V\}$ has 70% chance of being sampled from the distribution of *type* k, and 10% chance from being sampled from each of the other three types. The four *types* of distribution are uniformily random integers. The length (width) range for the types of distribution I to IV are, respectively, $[1, 50]$ ($[66, 100]$), $[66, 100]$ ($[1, 50]$), $[50, 100]$ ($[50, 100]$), and $[1, 50]$ ($[1, 50]$). *Possible sources of confusion:* In Berkey and Wang (1987), the term 'two-dimensional bin-packing problem' is used to describe what is now commonly referred to as Strip Packing Problem (i.e., packing into an strip

that is open ended in one dimension), the term is then adopted for 'a special case of packing into finite bins' which is the current meaning of the two-dimensional BPP. Also, in Martello and Vigo (1998), *seven* new classes are proposed, however, for this combined dataset only the first four classes (referred to as I, II, III, and IV in that work) are taken and they are referred to as classes VII, VIII, IX, and X in the context of this dataset of 10 classes (because the first six come from Berkey and Wang (1987)). The author believes this can be a source of confusion, as class VII in Martello and Vigo (1998) is *not* the class VII in the CLASS dataset.

OF1 and OF2 *Proposed in:* Oliveira and Ferreira (1990) *for the* unweighted G2KP (i.e., waste minization variant), and the *first known reference to the name adopted in this work is* Hifi and Roucairol (2001). *Other names:* not known, but possible, as the instances were just numbered in the proposing paper. *Characteristics:* The OF1–2 are part of a larger artificially generated dataset, but they were the only ones fully described in the body of the original paper and, as far as the author knows, the only ones adopted by later works. The generation procedure is clever but more complex than usual. Considering this thesis only employs two instances, the author do believe it is better just reproduce them than to describe the whole generation process. Both instances have $L = 40$, $W = 70$, and $n = 10$. The profit of each piece is set to their area. OF1 have $l = [5, 39, 9, 15, 16, 21, 14, 19, 36, 4]$, $w = [29, 9, 55, 31, 11, 23, 29, 16, 9, 22]$, and $u = [1, 4, 1, 1, 2, 3, 4, 3, 2, 2]$. OF2 have $l = [18, 10, 27, 18, 8, 4, 9, 19, 16, 16]$, $w = [22, 40, 13, 23, 29, 16, 47, 19, 13, 36]$, and $u = [2, 1, 3, 2, 4, 1, 1, 4, 2, 4]$.

STS1 to STS4 *Proposed in:* Tschöke and Holthöfer (1995) *for the* no-rotation and rotation G2KP, and the *first known reference to the name adopted in this work is* Alvarez-Valdés, Parajón and Tamarit (2002) (STS2 and STS4) and PackLib^2 (STS1 and STS3). *Other names:* the instances STS1 and STS3 are, in fact, cgcut1 and cgcut2 from Christofides and Whitlock (1977), the instances STS2 and STS4 (which were proposed by Tschöke and Holthöfer (1995)) are also called TH1 and TH2 by Fayard, Hifi and Zissimopoulos (1998). *Characteristics:* As STS1 and STS3 are cgcut1 and cgcut3 (both already described), this entry will focus exclusively on STS2 and STS4. Tschöke and Holthöfer (1995) mention the instances are artifical and fully specified in their appendix, but give no details of the generation procedure. The L, W, and n of the STS2 and STS4 are, respectively, $[55, 99]$,

$[85, 99]$, and $[30, 20]$. *Online repositories:* PackLib[2] <https://www.ibr.cs.tu-bs.de/alg/packlib/instances_problem_type.shtml>.

okp1 to okp5 *Proposed in:* Fekete and Schepers (1997) *for the* 2KP (i.e., non-guillotine G2KP), and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* the instances were artificially generated using the same schema than Beasley (1985b) "after applying initial reduction". The "initial reduction" seems to consist on a set of rules for reducing the pieces demand to the smallest value which does not affect the optimal objective value. The instances were fully described in the proposing paper. The original plate of all instances is 100x100, and the number of piece types in the five instances are 15, 30, 30, 33, and 29, respectively. The schema is the same as the one described in this list for the cgcut1–3 instances except that (i) the length is picked from $[1, L]$ (instead of the provisory piece area) and (ii) the demand vector was not handpicked but a random integer among 1, 2, and 3 (which may be changed by the "initial reduction", as mentioned above).

A1 to A5 *Proposed in:* Hifi (1997) *for the* weighted and unweighted G2KP (which is referred to as 'constrained two-dimensional cutting stock problem' in the paper), and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing papers; but note the similarly named instances A-1 to A-43 from Macedo, Alves and Carvalho (2010) have no relation to these instances. *Characteristics:* The instances are fully described in the paper, and their origin (real-world or artificial) is not mentioned. The instances A1 and A2 have arbitrary profits associated to the pieces, the A3, A4, and A5 do not (i.e., the piece area is used). The original plates range from 50x60 to 132x100, the piece demands range from 1 to 4, the average piece area of the first four instances is 699 (i.e., 26x27) and, for the fifth instance, 1107 (33x33).

CW1 to CW11 and CU1 to CU11 *Proposed in:* Fayard, Hifi and Zissimopoulos (1998) *for the* weighted and unweighted G2KP (CW means *constrained weighted* and CU mean *constrained unweighted*), and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* The CWi and CUi,

for $i = 1, \ldots, 11$, share L, W, l, w, and u; only p is distinct: in the CU instances $p_i = l_i \times w_i$, and in the CW instances p_i is a random integer in $[100, 1000]$; "the dimensions of the initial plate, the dimensions of the pieces and the number of pieces to cut m are uniformly taken in the integer intervals $[100, 1000]$, $[0.1 \times L, 0.7 \times W]$ and $[25, 60]$ respectively." (FAYARD; HIFI; ZISSIMOPOULOS, 1998). The pieces demand u_i were sampled using $max\{1, min\{10, random(0, \lfloor L/l_i \rfloor \times \lfloor W/w_i \rfloor)\}\}$. *Online repositories:* 2DPackLIB <http://or.dei.unibo.it/library/2dpacklib-2-dimensional-packing-pro <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>.

CHL1 to CHL7 *Proposed in:* Cung, Hifi and Cun (2000) *for the* G2KP, and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* "[...] the dimensions l_i and w_i of pieces to cut are taken uniformly from the intervals $[0.1L, 0.75L]$ and $[0.1W, 0.75W]$ respectively. The weight associated to a piece i is computed by $c_i = \lceil \rho l_i p_i \rceil$, where $\rho = 1$ for the unweighted case and $\rho \in [0.25, 0.75]$ for the weighted case. The constraints b_i, for $i = 1, \ldots, n$, have been chosen such that $b_i = min\{\rho_1, \rho_2\}$, where $\rho_1 = \lfloor L/l_i \rfloor \lfloor W/w_i \rfloor$ and ρ_2 is a number randomly generated in the interval $[1, 10]$" (CUNG; HIFI; CUN, 2000). Their c_i is what is referred to as p_i in this thesis (i.e., piece profit), the same can be said for b_i and u_i (i.e., piece demand). The L, W, and n must be provided, and for CHL1–7 they are [132, 62, 157, 207, 20, 130, 130], [100, 55, 121, 231, 20, 130, 130] and [30, 10, 15, 15, 10, 30, 35], respectively. *Online repositories:* PackLib^2 <https://www.ibr.cs.tu-bs.de/alg/packlib/instances_problem_type.shtml>, <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>.

Hchl1, Hchl2, Hchl9, and Hchl3s to Hchl8s *Proposed in:* Cung, Hifi and Cun (2000) *for the* weighted and unweighted G2KP, and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* The generation procedure is the same described in item CHL1–CHL7. The suffix 's' is used to indicate that the pieces have a profit value equal to their area (as in A1s, A2s, 2s, ...). However, differently of the other suffixed instances, which had an earlier version without the suffix and with an arbitrary profit vector, there does not seem to exist instances Hchl3–Hchl8. The L, W, n for the instances (in the order they are numbered) are [130, 130, 127, 127, 205, 253, 263, 49, 65], [130, 130, 98, 98,

223, 244, 241, 20, 76], and [30, 35, 10, 10, 25, 22, 40, 10, 35], respectively. *Online repositories:* <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>.

A1s, A2s, 2s, 3s, STS2s, STS4s, and CHL1s to CHL4s *Proposed in:* Cung, Hifi and Cun (2000) *for the* unweighted G2KP, and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing papers), however, in the text pf the original paper there was a single space separating the instance name from the 's' which was supressed by all subsequent works. *Characteristics:* "The instances 2 s–3 s, A1 s–A2 s, STS2 s–STS4 s and CHL1 s–CHL4 s represent exactly the instances 2–3 , A1–A2, STS2–STS4 and CHL1–CHL4, respectively for which the profit of each piece is represented by its area.". (CUNG; HIFI; CUN, 2000) The original version of each instance (i.e., without the 's') is described by other items of this list. The instances 2 and 3 refer to the alternative name to for the cgcut2 and cgcut3.

T1a to T7e and N1a to N7e *Proposed in:* Hopper (2000) *for the* Strip Packing Problem with the guillotine constraint (T instances, but see below) and without the guillotine constraint (N instances), and the *first known reference to the name adopted in this work is* the proposing thesis. *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* All instances of both datasets have a strip width of 200, and a (supposedly) known optimal height of 200 (with no waste). Each dataset has 35 instances which are divided into categories 1 to 7, each category has five distinct instances indicated by the last character of the instance name: a, b, c, d, e; and these five instances differ from each other only because the seed given to the random number generator. Each category define the same total number of pieces in both datasets, respectively: 17, 25, 29, 49 73, 97, 199. All instances are fully described in the appendix of Hopper (2000). The generation procedure is also described by the proposing thesis and consists mostly of starting with a 200x200 plate, and recursively making guillotine cuts to the plates (T instances), or recursively dividing the plates into the five plates of the basic non-guillotinable pattern (N instances). In Section 7.6, a proof by exhaustion shows that T1a is impossible to pack into a guillotine pattern of 200x200, and there is strong evidence empirical evidence of this impossibility for other T instances. The author believes that both dataset T and N were created using the procedure to create non-guillotinable patterns.

C1-p1 to C7-p3 *Proposed in:* Hopper and Turton (2001) *for the* Strip Packing Problem (without the guillotine constraint and allowing piece rotation) and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* while the instances were named by the proposing paper, some papers refer to them by 'HT' followed by a number or another identifier(GRANDCOLAS; PINTO, 2010; FLESZAR, 2016). *Characteristics:* The dataset of 21 instances is divided into categories 1 to 7 (the *C* comes from category) each with problems 1 to 3 (the *p* comes from problem). Each category defines the total number of pieces, a strip width, and the optimal height (known at generation). The number of pieces are 16 or 17 (C1), 25, 28 or 29, 49, 72 or 73, 97, and 196 or 197 (C7). The strip widths are: 20, 40, 60, 60, 60, 80 and 160. The optimal heights are: 20, 15, 30, 60, 90, 120, and 240. Each of the 21 instances is described in its entirety inside the proposing paper. The optimal solution is always a pattern with no waste, and it " is achieved by packing the rectangles in the order they are stated in the tables using the BLF routine." (HOPPER; TURTON, 2001). The proposing paper does not give details of the generation process. In this thesis, the experiments show that, for some instances, it is impossible to obtain the known optimal solution while enforcing guillotine cuts and disallowing rotation; however, when rotation is allowed, the optimal solution was found for every instance which the runs did not timeout.

APT10 to APT49 *Proposed in:* Alvarez-Valdés, Parajón and Tamarit (2002) *for the* G2KP (unconstrained unweighted and weighted, and constrained weighted and unweighted) and the *Other names:* not known and improbable (the instances were named by the proposing paper). *Characteristics:* The instances APT10 to APT29 are unconstrained, L and W are sampled from $[1500, 3000]$ and the number of pieces types is sampled from $[30, 60]$. The instances APT30 to APT49 are constrained, L and W are sampled from $[100, 1000]$, the number of pieces types is sampled from $[25, 60]$, and the demand of each piece i comes from $minu_i^1, u_i^2$ where $u_i^1 = \lfloor L/l_i \rfloor \times \lfloor W/w_i \rfloor$ and u_i^2 is sampled from $[1, 10]$. For all intances, the piece lengths are sampled from $[0.05L, 0.4L]$, and the piece widths are sampled from $[0.05W, 0.4W]$. The instances APT10 to APT19 (unconstrained), as well as APT30 to APT39 (constrained), are unweighted, this is, the profit of each piece is equal to its area. The instances APT20 to APT29, as well as APT40 to APT49, are weighted, and the profit of each piece i is sampled from $[0.25l_iw_i, 0.75l_iw_i]$.

CJCM *Proposed in:* Clautiaux, Carlier and Moukrim (2007), but sometimes misatributted to Clautiaux et al. (2008), *for the* ortogonal packing problem (this is, the non-guillotined G2OPP) and the *first known reference to the name adopted in this work is* Côté and Iori (2018) which explicitly refer to the dataset as CJCM, some citation styles abbreviate the authors as 'CJCM', so there is a previous work that referred to the instances as 'the instances from [CJCM08]' (BELOV; ROHLING, 2009). *Other names:* the author is also aware of the name 'CCM' employed by (FLESZAR, 2016); as the dataset was not named by its authors and the instances were identified by their three unique attributes (see below) it is possible other names exist. *Characteristics:* Unfortunately, Clautiaux, Carlier and Moukrim (2007) points to Hopper and Turton (2002) for details on the instance generation method and the author was not able to find a copy of the latter. Besides that reference, the proposing paper tell us that "The idea is to obtain both feasible and non-feasible problem instances. The first step of the algorithm consists in randomly generating a set of values whose sum is equal to $(1-\epsilon)WH$. These values are the areas of the items in the created instance. Then the values are factorized to get the width and the height of the items." (CLAUTIAUX; CARLIER; MOUKRIM, 2007). In this thesis, the name of individual instances follow the pattern Eϵfn, in which ϵ are two digits indicating $(LW - \sum_{i \in \bar{J}} l_i w_i u_i)/100$ (i.e., guaranteed waste percentage if feasible), f is the feasibility status as determined by the methods employed in the proposing paper ('F' for feasible, 'N' for unfeasible, and 'X' for not solved by any of the three methods employed within the time limit of 15 minutes for each), and n is the total number of pieces ($\sum_{i \in \bar{J}} u_i$).). The selection criteria for the 42 instances of the dataset is not entirely clear: the distribution of n in the range $[10, 23]$ seems arbitrary, and instances with the same ϵ and n only exist if f (assessed by the experiments) is distinct. Consequently, it seems like a larger dataset was first generated, and then 42 instances were selected from it based on the results of the experiments. The original plate dimensions are always 20x20, and the piece types are heterogeneous ($u_i = 1$ for the vast majority of the piece types, going up to $u_i = 4$ for five piece types in four distinct instances).

A-1 to A-43 *Proposed in:* Macedo, Alves and Carvalho (2010) *for the* two-staged G2CSP and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing paper); but note the similarly named instances A1 to A5 from Hifi (1997)

have no relation to these instances. *Characteristics:* "[...], two sets of real instances from the furniture industry, set A and [...]" (MACEDO; ALVES; CARVALHO, 2010, p. 7). The instances are highly heterogeneous, and their numbering seem to have no relation with any of their characteristics. The most consistent characteristic are the dimensions of the original plates, which are either 2550x2100 (31 instances), 2750x1220 (11 instances), or 2470x2080 (1 instance). A table summarising their characteristics can be found in the proposing paper. *Online repositories:* 2DPackLib <http://or.dei.unibo.it/library/2dpacklib>.

P1_*, P2_*, P3_*, and P4_* *Proposed in:* Velasco and Uchoa (2019) *for the* rotation and no-rotation G2KP, and the *first known reference to the name adopted in this work is* the proposing paper. *Other names:* not known and improbable (the instances were named by the proposing papers). *Characteristics:* There is a total of 80 instances, each combination of $i \in \{1, 2, \ldots, 5\}$ (random number generator seed) and $n \in \{25, 50\}$ for each of the following 8 triples of *class*, L, and W: $(1, 100, 200)$, $(1, 100, 400)$, $(2, 200, 100)$, $(2, 400, 100)$, $(3, 150, 150)$, $(3, 250, 250)$, $(4, 150, 150)$, $(4, 250, 250)$. The instance names follow the pattern P*class*_L_W_n_i. The pieces of instances with $n = 25$ are a subset of the pieces in instances with $n = 50$ (i.e., the 'first half'). The piece demands are randomly picked from the integer uniform distribution $[1, 9]$, and the piece profits are the piece area multiplied by a real number randomly picked from the continuous distribution between 0.5 and 1.5. The classes 1–3 have piece lenghts randomly picked from the integer uniform distribution $[5, 40]$, and piece widths from $[10, 80]$. In class 4, half the pieces have their length and width defined in the same way as classes 1–3, and the other half uses $[10, 80]$ for length, and $[5, 40]$ for width, i.e., the distributions are switched between dimensions. *Online repositories:* 2DPackLIB <http://or.dei.unibo.it/library/2dpacklib-2-dimensional-packing-problems-library>.

CW*_M2, 4, 8 *Proposed in:* this thesis (but it is a direct adaptation of the CW1–CW11 instances from Fayard, Hifi and Zissimopoulos (1998)) *for the* G2MKP and the *first known reference to the name adopted in this work is* this thesis (the dataset as a whole may be referred to as CW_M). *Other names:* none. *Characteristics:* each instance CWk_Mm is exactly the same as the original CWk instance except it has m original plates available instead of just one. To retain the selection aspect of the problem (i.e., *which* pieces will be packed, not only *how* they will be packed), the

author chose to only have a CWk_Mm instance if $(\sum_{i \in \bar{J}} l_i w_i u_i) \geq 2mLW$. This is the reason CW1_M8 to CW5_M8 do not exist: each instance from CW1 to CW5 has a summed area of their pieces lower than the summed area of 2×8 copies of their respective original plates.

A-*_M2, A-*_M4, A-_M8 *Proposed in:* this thesis (but it is a direct adaptation of the A-1 to A-43 instances from Macedo, Alves and Carvalho (2010)) *for the* G2MKP and the *first known reference to the name adopted in this work is* this thesis. *Other names:* none. *Characteristics:* each instance A-k_Mm is exactly the same as the original A-k instance except it has m original plates available instead of just one. To retain the selection aspect of the problem (i.e., *which* pieces will be packed, not only *how* they will be packed), the author chose to only have a A-k_Mm instance if $(\sum_{i \in \bar{J}} l_i w_i u_i) \geq 2mLW$. The set of instances A-1 to A-43 is very heterogeneous, and the numbering seem to be arbitrary, so this leads to an equally arbitrary subset of them respecting the criteria for $m \in \{2, 4, 8\}$. For the sake of completeness, the numbers of the original instances that have $m = 2$ counterparts are: 2, 3, 5, 7, 9, 11–21, 23–29, 31–38, and 40–43 (i.e., 35 of the original 43 instances); of these the following 26 original instances have $m = 4$ counterparts, they are: 2, 5, 7, 9, 11–16, 18–21, 23–25, 27–29, 32, 33, 35, 38, 40, and 41; and, finally, of these the following 16 original instances have $m = 8$ counterparts: 2, 9, 11, 14–16, 18–21, 24, 27, 29, 32, 33, and 38. This totalises $35 + 26 + 16 = 77$ instances in this new dataset.

FMT59 *Proposed in:* Furini, Malaguti and Thomopulos (2016) (no instance is new so it was not proposed but grouped) *for the* weighted, unweighted, constrained, and unconstrained G2KP and the *first known reference to the name adopted in this work is* this thesis. *Other names:* none. *Characteristics:* the dataset contains 37 unweighted instances and 22 weighted instances (59 in total) and comprises the entirety of some literature datasets as well as subsets of others. Every instance come from another dataset described in this list. The unweighted instances are: W, wang20 (WANG, 1983), gcut1 to gcut12 (BEASLEY, 1985a), OF1, OF2 (OLIVEIRA; FERREIRA, 1990), A3 to A5 (HIFI, 1997), CU1, CU2 (FAYARD; HIFI; ZISSIMOPOULOS, 1998), 2s, 3s, A1s, A2s, STS2s, STS4s, CHL1s, CHL2s, CHL5, CHL6, CHL7, Hchl3s, Hchl4s, Hchl6s, Hchl7s, Hchl8s(CUNG; HIFI; CUN, 2000). The weighted instances are: cgcut1 to cgcut3 (CHRISTOFIDES;

WHITLOCK, 1977), okp1 to okp5 (FEKETE; SCHEPERS, 1997), HH (HIFI, 1997), 2, 3 (CHRISTOFIDES; WHITLOCK, 1977), A1, A2 (HIFI, 1997), STS2, STS4, CHL1, CHL2, CW1 to CW3, Hchl2 and Hchl9 (CUNG; HIFI; CUN, 2000). The dataset was assembled from an assortment of datasets from the OR library (wang20, gcut1 to gcut12, cgcut1 to cgcut3, okp1 to okp5) and the already mixed dataset employed in Hifi and Roucairol (2001) (all the remaining instances). The author do not suggest employing this dataset as it is in future works, unless a comparison with a work that has already employed it is needed. The reason for this recommendation is that, because the literature adopted different names for the same instances, some instances of the dataset are just duplicates with different names. This is the case for instances 2 and 3 which are the same as cgcut2 and cgcut3. Taking into account the literature wang20 and W should be the same instance but the instances obtained by the author have distinct demand values for some of the pieces (all the remaining instance characteristics are the same for all pieces). The weighted instances A1, A2, 2, 3, STS2, STS4, CHL1, and CHL2 have an unweighted version of themselves in the same dataset; the unweighted alternatives have the same name but are suffixed with an 's'.

Easy18 A subset of the dataset FMT59 defined in this work. Its purpose is to reduce the number of runs needed before discarding a formulation from further consideration. The dataset contains: cgcut1 to cgcut3, gcut1 to gcut12, OF1, OF2, and wang20. See FMT59 for the origin of each instance, and the rest of the list for a description of their characteristics.

APPENDIX B — RESUMO EXPANDIDO

O problema principal desse trabalho é o Problema da Mochila 2D Guilhotinada com cortes ortogonais (e irrestritos), demanda limitada, estágios ilimitados, e sem rotação. Essa variante específica é referida doravante como PM2G. O PM2G é um problema fortemente NP-difícil (KORF, 2003; DOLATABADI; LODI; MONACI, 2012). Esse trabalho também examina um tipo específico de corte restrito, três problemas distintos relacionados ao PM2G, e a variante que permite a rotação das peças (em todos problemas estudados). Os três problemas distintos mencionados acima são o Problema das Múltiplas Mochilas (PMM), o Problema de Empacotamento Ortogonal (PEO), e o Problema de Corte de Estoque (PCE). Esse trabalho foca na obtenção de soluções ótimas para esse problema através de Programação Linear Inteira Mista (PLIM).

Uma instância do PM2G consiste de: um retângulo de comprimento L e largura W (aqui chamada de *placa original*); um conjunto de retângulos $\bar{J}$ (doravante chamados de *peças*) onde cada retângulo $i \in \bar{J}$ tem um comprimento l_i, uma largura w_i, um lucro p_i, e uma demand u_i. O PM2G procura maximizar o lucro das peças obtidas pelo corte da placa original. O termo *guillotinado* significa que cada corte sempre vai de um lado da placa até o lado oposto da mesma; um corte nunca para ou começa no meio de uma placa.

Problemas de corte guilhotinado são de interesse da indústria, especialmente da indústria da madeira (YANASSE; MORABITO, 2008; MORABITO; BELLUZZO, 2007) e do vidro (CLAUTIAUX et al., 2019; PARREÑO; ALONSO; ALVAREZ-VALDES, 2020), em geral devido a limitações do maquinário. Existe uma literatura vasta e em crescimento no assunto como é evidenciado por Iori et al. (2020) e por Russo et al. (2020).

Esse trabalho foca em PLIM como método de solução (ao invés de métodos *ad hoc*) porque a adaptabilidade amplifica o valor de quaisquer melhoramentos descobertos. Uma formulação PLIM melhor significa: um método de solução melhor para os vários (já mencionados) problemas relacionados; uma melhor relaxação contínua para computar um chute otimista do valor da função objetivo de todas essas variantes (alguns algoritmos *ad hoc* usam solucionadores PLIM para computar esses limites); não somente um melhor método exato mas também uma base melhor para heurísticas e procedimentos que suportam interrupção a qualquer momento; a capacidade automática de se beneficiar de paralelização, decomposição automática de problemas, e heurísticas do solucionador; e, finalmente, melhor envelhecimento do método através dos anos por meio da tendência atual de processadores de múltiplos núcles e o constante avanço na performance dos

solucionadores.

As contribuições dessa tese incluem:

- uma formulação PLIM baseada em uma formulação prévia do estado da arte, a prova da sua corretude, e evidência empírica do melhoramento da performance;
- uma nova forma de empregar uma propriedade já conhecida (normalização do tamanho da placa) para tanto a formulação original quanto a formulação melhorada, e evidência empírica do impacto positivo deste uso original na performance de ambas formulações;
- novos limitantes inferiores e superiores, assim como novos valores ótimos, para várias das instâncias difíceis propostas recentemente Velasco and Uchoa (2019);
- uma comparação direta com formulações recentes da literatura enfatizando os pontos fracos e fortes de cada;
- uma adaptação da formulação proposta para três problemas (PMM2G, PEO, e PCE2G) e resultados empíricos sobre conjuntos de dados da literatura para servir como uma base para futuras comparações entre formulações;
- a hibridização da formulação proposta (com a formulação de Silva, Alvelos and Carvalho (2010)) que tem sucesso moderado em reduzir ainda mais o tempo de execução para instâncias em que a maior parte to tempo é dispendida na fase de B&B (*branch and bound*) por meio da proibição de certas simetrias.

Para tal, o autor reimplementou uma formulação do estado da arte e um procedimento de precificação usado por ela.

A primeira formulação PLIM feita especificamente para o PM2G foi proposta por Furini, Malaguti and Thomopulos (2016). A formulação é classificada como uma extensão da formulação *one-cut* de Dyckhoff (1981) para o Problema de Corte de Estoque unidimensional. Entretanto, a formulação de Silva, Alvelos and Carvalho (2010) pode ser vista como uma etapa intermediária entre essas duas: esta já havia estendido a formulação *one-cut* para duas dimensões mas não havia alterado o problema do PCE2G para o PM2G e era limitada a dois estágios ou três estágios restritos. Uma versão estendida de Furini, Malaguti and Thomopulos (2016) aparece em Thomopulos (2016) (uma tese de doutorado), e um prelúdio a ela aparece em Furini and Malaguti (2016). A formulação tem tamanho pseudo-polinomial, $O((L+W) \times L \times W)$ variáveis e $O(L \times W$ restrições,

além da sua relaxação prover limitantes melhores que a formulação em Messaoud, Chu and Espinouse (2008).

Nesse trabalhi, o autor propõe uma formulação melhorada baseada na formulação de Furini, Malaguti and Thomopulos (2016) mencionada acima. Uma vantagem significativa desse melhoramento é evitar a enumeração de quaisquer cortes depois da metade de uma placa. Essa vantagem aparece em muitos trabalhos desde Herz (1972).

Considerando as 59 instâncias usadas nos experimentos da formulação em que o autor se inspirou, e somando os valores para todos os modelos gerados, a formulação proposta tem apenas uma pequena fração das variáveis e restrições do modelo original (respectivamente, 3.07% e 8.35%). A formulação melhorada soluciona todas as 59 instâncias em cerca de 4 horas enquanto a formulação original soluciona 53 em 12 horas (as outras 6 instâncias não são solucionadas dentro do limite de 3 horas por instância). Nós integramos, em ambas formulações, uma estrutura de precificação proposta para a formulação original; a formulação melhorada mantém uma vantagem significativa nessa situação. Em um conjunto de 80 instâncias difíceis recentemente proposto, a formulação melhorada (com e sem a estrutura de precificação) encontrou: 22 soluções ótimas para o problema com cortes irrestritos (5 já conhecidas, 17 novas); 22 soluções ótimas para o problema com cortes restritos (todas novas para o problema e nenhuma é a mesma que do problema de cortes irrestritos); melhores limitantes inferiores para 25 instâncias; melhores limitantes superiores para 58 instâncias. Considerando outras formulações para o problema na literatura, a formulação proposta apresenta tempos de execução menores, e prova a otimalidade para mais instâncias. Somente nos conjuntos de instâncias em que nenhuma formulação solucionou instância alguma é que a formulação proposta falhou em encontrar boas soluções primais enquanto outras formulações obtiveram êxito. A formulação proposta somente falhou em obter soluções de boa qualidade nos conjuntos de instâncias em que nenhuma formulação conseguiu solucionar instância alguma. Nesses conjuntos de dados, outras formulações obtiveram boas soluções primais mesmo não sendo capazes de solucionar instância alguma.