Mathematical Programming Computation (2022) 14:673–697
https://doi.org/10.1007/s12532-022-00222-4

FULL LENGTH PAPER

Enhanced formulation for the Guillotine 2D Cutting Knapsack Problem

Henrique Becker[1] · **Olinto Araújo**[2] · **Luciana S. Buriol**[1]

Received: 10 October 2020 / Accepted: 20 April 2022 / Published online: 12 July 2022
© The Author(s) 2022

Abstract
We advance the state of the art in Mixed-Integer Linear Programming formulations for Guillotine 2D Cutting Problems by (i) adapting a previously-known reduction to our preprocessing phase (plate-size normalization) and by (ii) enhancing a previous formulation (PP-G2KP from Furini et alli) by cutting down its size and symmetries. Our focus is 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. The code is available. Concerning the set of 59 instances used to benchmark the original formulation, the enhanced formulation takes about 4 hours to solve all instances while the original formulation takes 12 hours to solve 53 of them (the other six runs hit a three-hour time limit each). We integrate, to both formulations, a pricing framework proposed for the original formulation; the enhanced formulation keeps a significant advantage in this situation. Finally, in a recently proposed set of 80 harder instances, the enhanced formulation (with and without the pricing framework) found: 22 optimal solutions (5 already known, 17 new); better lower bounds for 25 instances; better upper bounds for 58 instances.

Keywords Combinatorics · Symmetry breaking · Pseudo-polynomial · Formulation

Mathematics Subject Classification 68R05 · 68U99 · 05D99 · 52B99

✉ Henrique Becker
hbecker@inf.ufrgs.br

Olinto Araújo
olinto@ctism.ufsm.br

Luciana S. Buriol
buriol@inf.ufrgs.br

[1] Federal University of Rio Grande do Sul (UFRGS), Av. Bento Gonçalves, Porto Alegre, RS 9500, Brazil

[2] Federal University of Santa Maria (UFSM), Av. Roraima, Santa Maria, RS 1000, Brazil

1 Introduction

The problem we focus on in this work is the Guillotine 2D Knapsack Problem with orthogonal (and unrestricted) cuts, constrained demand, unlimited stages, and no rotation. We will refer to this specific variant as G2KP. The G2KP is a strongly NP-hard problem [25]. The work also focuses on obtaining optimal solutions for this problem through Mixed-Integer Linear Programming (MILP). We propose two simple but effective enhancements regarding a state-of-the-art MILP formulation for the G2KP (which may also benefit some closely related problem variants).

1.1 Explanation of the problem and some close variants

An instance of the G2KP consists of: a rectangle of length L and width W (hereafter called *original plate*); a set of rectangles $\bar{I}$ (also referred to as *pieces*) where each rectangle $i \in \bar{I}$ has a length l_i, a width w_i, a profit p_i, and a demand u_i. We assume, 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 go from one side of a plate to other; a cut never stops or starts from the middle of a plate. We cut the original plate into intermediary plates $j \in J$, $J \supseteq \bar{I}$, which we further cut following the same rule.

If we do not cut a plate further, then it is either: thrown away as trim/waste for no profit; or, if it has the same size as a piece, it may also be sold by the piece profit value. *Orthogonal cuts* are always parallel to one side of a plate (and perpendicular to the other). Consequently, any intermediary plate j is always a rectangle, and have a well-defined l_j and w_j. *Unrestricted cuts* mean we are allowed to make horizontal (vertical) cuts different from the length (width) of a piece. In constrast, restricted cuts means 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), we create and employ the term *position-only restricted* to keep the reader aware of what we mean. Solving the position-only restricted problem exactly is a costly but high-quality primal heuristic for the G2KP.

Constrained demand means we can sell at most u_i copies of piece i. The G2KP with *unconstrained demand* is not strongly NP-hard but weakly NP-hard instead; exact algorithms of pseudo-polynomial time complexity exist [2]. Consequently, if $u_i \geq \beta_i : \forall i \in \bar{I}$, 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. We avoid this kind of instances in our experiments. *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, a 2-staged G2KP has all cuts in some orientation before any cuts in the other orientation. 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. The *no-rotation* qualifier means we never switch

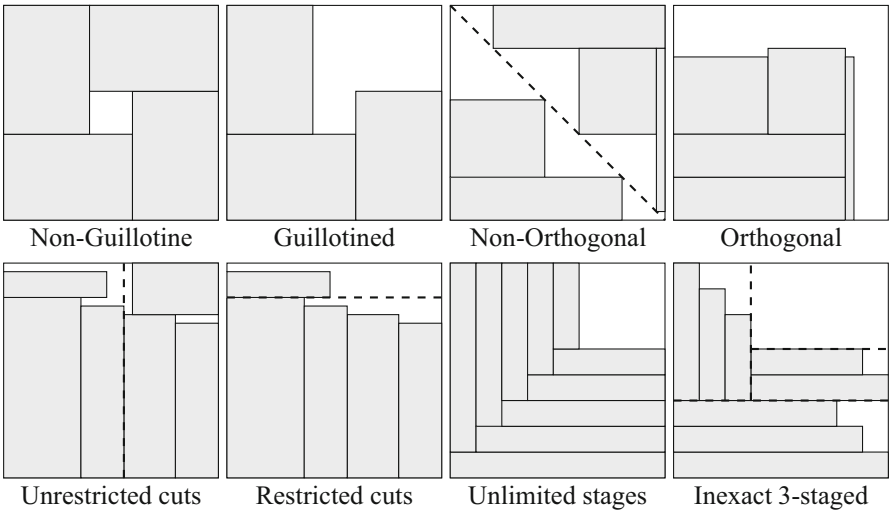

Fig. 1 Examples of valid patterns for most of the discussed problem variants. In *Non-Orthogonal*, *Unrestricted cuts*, and *Restricted cuts*, the dashed line indicate the first cut of the pattern. In *Inexact 3-staged*, the dashed line separates the three stages

length and width during the cutting process; especially, we cannot sell a plate j as a piece of length w_j and width l_j.

If we further qualify the G2KP, we only mean to discard the qualifiers above that directly conflict with the extra qualifiers, if any. For example, if we refer to the *unconstrained G2KP*, then we meant 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 may help to understand some of the discussed characteristics.

While our work focuses on this specific problem, the enhanced formulation we present may be readily adapted to, at least, the Guillotine 2D version of the following problems: the Cutting Stock Problem (and the Bin Packing Problem); the Strip Packing Problem; the Multiple Knapsack Problem; the Orthogonal Packing Problem; and the variant allowing rotation for all previously mentioned problems. See [14] for more details. We do not define or further discuss these problems or variants in this work.

1.2 Motivation

Guillotine cutting problems are of interest of the industry, especially the wood [22, 31] and glass cutting industries [9, 23], often because of machinery limitations. 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 [23] for more details on this challenge. The vast and growing literature on the subject, pointed out by two recent surveys [17, 25], is also evidence of such interest.

We focus on MILP as the solving method (instead of *ad hoc* solutions) because its adaptability amplifies the value of any enhancements we obtain.

1.3 Contributions and paper outline

The main contributions of this work are: 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 straightforward adaptation of a previously known reduction procedure for both the original and the enhanced formulations, and empirical evidence of its positive impact on their performance; finally, we present new upper and lower bounds, as well as optimal values, for many recently proposed hard instances from [30]. For such, we reimplement a state-of-the-art MILP formulation and an optional pricing procedure used by it. This reimplementation allows us to compare both approaches fully. For reproducibility, the exact version of the code employed in this paper is available at [3]. However, we suggest using the better documented and maintained master branch (https://github.com/henriquebecker91/GuillotineModels.jl) if perfect reproduction is not necessary.

We organise the rest of the paper the following way: Sect. 2 analyses how our work interacts with the pre-existing literature; Sect. 3 introduces some mathematical concepts and explains the reduction we adapted from the literature; Sect. 4 describes our enhanced formulation and briefly explains how it differs from the state-of-the-art formulation it is based on; Sect. 6 presents our experiments and the empirical results we derive from them; Sect. 7 delivers our conclusions and suggests future work.

2 Related work

We do not intend to provide a full overview of the literature, instead we: refer to surveys; discuss only closely related works and how they interact with our contributions; and opportunely point out missing connections between related works.

Two relevant surveys have come out recently. [17] catalogues exact methods and relaxations for 2D cutting problems including guillotine problems. [25] reviews the literature of our particular problem at length – there G2KP is referred to as Constrained 2D Cutting or C2DC. Moreover, [25] points out three strategies employed by previous exact solving methods which cause loss of optimality, i.e., these methods cannot be considered exact anymore. Our work does not employ any of these three strategies. One of these strategies is a dominance rule that is valid for the unconstrained case but not for the constrained case. In 1972, [16] proposed a dominance rule for the G2KP with unconstrained demand 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 [5] in 2008. The problem considered by [5] is the Strip Packing Problem[1],

[1] 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.

but adapting the formulation to the knapsack variant would not change its fundamentals. Previously, [18] had proposed two MILP formulations for 2-staged G2KP. As noted by [4], 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 [5] 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 [5] seems to have been simultaneously proposed by [24].

The first MILP formulation specifically for the G2KP was proposed by [14] in 2016. An extended version of [14] appears in [29] (a PhD thesis). 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 [5]. It was the first formulation able to solve medium-sized instances of the literature. Besides the formulation, [14] proposes two reductions and one pricing procedure; all of these are reimplemented by our 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 [27].

In this work, we propose an enhanced formulation based on the one from [14] mentioned above. A significant advantage of our enhancement is to avoid the enumeration of any cuts after the middle of a plate. This advantage appears in many works since [16]. Recently, [11] adapted a formulation for the one-dimensional Cutting Stock Problem to obtain this same advantage. However, the way [11] changes their formulation to obtain this advantage is not the same as our approach.

The most recent MILP formulations for the G2KP come from three works by Martin et alii [19–21]. These formulations are compared against the formulation of [14]. We base our enhanced formulation on [14] and also compare against it. The formulations of [19–21] have a looser relaxation bound compared to [14], but perform better than [14] in instances for which [14] has a much larger number of variables. Considering the instances used in [14], our enhanced formulation dominates the formulation of [14]. Our formulation also dramatically improves the running times of instances in which the formulation of [14] performed worse than [19–21] (e.g., the gcut1–gcut12 instances). Consequently, while it may be interesting for completeness sake, we do not compare against the formulations proposed in [19–21].

3 Notation, discretization, and plate-size normalization

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 [8] and [16], 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 *discretizations*. The most common way of computing these discretizations are 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 [14] and our enhanced formulation 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[2], 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 (discretization) considered are the canonical dissections of [16], hereafter referred to as *normal cuts* instead. We acknowledge the existence of stricter discretizations: the raster points of [26, 28], the regular normal patterns of [7] (named this way by [10]), and the Meet-in-the-Middle (MiM) of [10]. The reasons for our choice of discretization are numerous: it works well with the *Plate-Size Normalization* procedure we describe below; it is the same discretization employed by [14] (from which we base our enhanced formulation on); The main gain of MiM is reducing the number of cut positions after the middle of a plate, which our enhanced formulation already discards anyway; the regular normal patterns compute a distinct subset-sum for each pair of plate and piece, which we consider excessive (there may exist hundreds of thousands of intermediary plate possibilities); finally, the raster points complicate our proofs and our *Plate-Size Normalization* weakens its benefits.

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

$$N_{jo} = \begin{cases} \{q : 0 < q < l_j; \quad \exists n_i \in [0\,..\,u_i], \forall i \in I_j, q = \sum_{i \in I_j} n_i l_i\} & \text{if } o = h, \\ \{q : 0 < q < w_j; \; \exists n_i \in [0\,..\,u_i], \forall i \in I_j, q = \sum_{i \in I_j} n_i w_i\} & \text{if } o = v. \end{cases} \tag{1}$$

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).

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

The goal of the *Plate-Size Normalization* procedure we propose 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 similarly exploited by [1] and by [12]. We state the property as:

Proposition 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). We do not replicate any proof here. We can then define:

Definition 1 The length of a plate j is considered normalized if, and only if, $l_j = l'_j$. The analogue is valid for the width. The size of a plate is normalized if, and only if, both its length and its width are normalized.

The *Plate-Size Normalization* procedure we propose consists only of replacing every non-size-normalized plate enumerated by their normalized counterpart. The number of distinctly-sized plates diminishes because the procedure replaces many plates of distinct but similar dimensions by a single plate. The only extra effort added by *Plate-Size Normalization* 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 our implementation, we 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 our worst-case complexity increases, the time spent decreased because the actual number of plates (denoted in the complexity by $O(LW)$ became 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 Normalization* (no extra effort required).

Remark 1 If a normal cut divides a size-normalized plate, then the dimension perpendicular to the cut, in the first child, is normalized. The dimension parallel to the cut in the first child, and both dimensions of the second child, are not guaranteed to be normalized.

Example 1 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 normalized. The plate length matches stacking the three copies of the first piece. The plate width matches laying side-by-side the other two pieces. An horizontal cut at length 7 is a normal cut

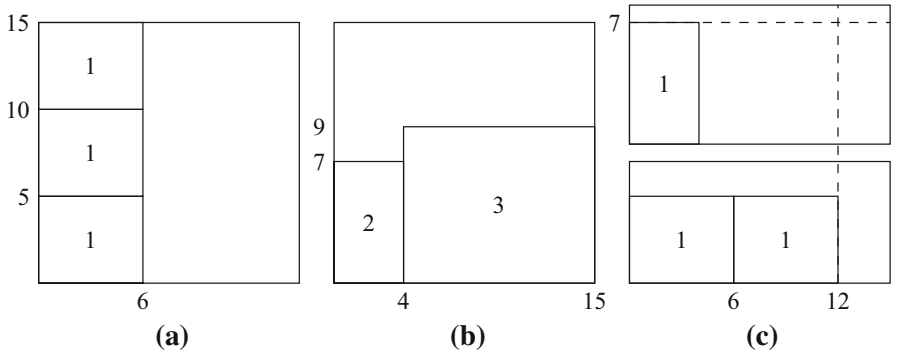

Fig. 2 Diagram of 1: **a** the three copies of the first piece stacked; **b** the second and third pieces side-by-side; **c** both children of a horizontal normal cut over a normalized plate are not normalized themselves

because it matches the length of the second piece. If the cut is done, the width of both children is not normalized anymore, nor is the length of the second child. The width of both children is not normalized 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 normalized because the largest length a valid packing inside the second child may reach is 7. The dimensions of both children may be normalized 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.

4 Our changes to Furini's model

The formulation proposed in [14] is elegant: the pieces are just intermediary plates that may be sold. Our contribution consists of changes to both the preprocessing step and to the formulation. These changes significantly reduce the number of variables. Differently, these changes deepen the distinction between plates and pieces and, consequently, may be regarded as sacrificing some elegance for performance. The essentials of the formulation remain the same and, for this reason, we consider the model presented here as an enhanced model, not an entirely new model.

The cut enumeration in [14] excludes some symmetric cuts; that is, if two different cuts create the same set of two child plates, then the symmetric cut in the second half of the plate may be ignored. Differently, [8] disregards *all* cuts after the middle of the plate because of symmetry. If [14] would do the same as [8] it could become impossible to trim a plate to the size of a piece. For example, if there was a piece with 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 the plate. The goal of our changes is to reduce the number of cuts (i.e., model variables) by getting closer to the symmetry-breaking rule used in [8] without loss of optimality.

4.1 The enhanced formulation

Our changes to the formulation are restricted to replacing the set of integer variables $y_j, i \in \bar{J}$, 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 the original formulation, y_i denoted the number of times a plate i was sold as the piece i, in this case, the plate always had the exact size of the piece. Our *extraction variables* e_{ij} denote a piece i was extracted from plate j, which size may differ from the size of the piece. The exact definition of set E is discussed over Sect. 4.2; for the purpose of presenting the formulation, our intuitive definition of e_{ij} just above is enough. For convenience, we also define $E_{i*} = \{j : \exists\,(i, j) \in E\}$ and $E_{*j} = \{i : \exists\,(i, j) \in E\}$. 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 parameter a is a byproduct of the plate enumeration process. 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 $q \in Q_{ko}$. The description of this parameter in [14] has a typo, as pointed out by [19]: "[...] there is a typo in their definition of parameter a^o_{qkj}, as the indices j and k seem to be exchanged.".

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 $q \in Q_{jo}$; while the value of e_{ij} is the number of sold pieces of type $i \in \bar{J}$ that were extracted from plates of type $j \in J$. 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.

$$\text{max.} \quad \sum_{(i,j)\in E} p_i e_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_{o\in O}\sum_{q\in Q_{jo}} x^o_{qj} + \sum_{i\in E_{*j}} e_{ij} \leq \sum_{k\in J}\sum_{o\in O}\sum_{q\in Q_{ko}} a^o_{qkj} x^o_{qk} \qquad \forall j \in J, j \neq 0, \tag{3}$$

$$\sum_{o\in O}\sum_{q\in Q_{0o}} x^o_{q0} + \sum_{i\in E_{*0}} e_{i0} \leq 1 \qquad , \tag{4}$$

$$\sum_{j\in E_{i*}} e_{ij} \leq u_i \qquad \forall i \in \bar{J}, \tag{5}$$

$$x^o_{qj} \in \mathbb{N}^0 \qquad \forall j \in J, o \in O, q \in Q_{jo}, \tag{6}$$

$$e_{ij} \in \mathbb{N}^0 \qquad \forall (i, j) \in E. \tag{7}$$

The objective function maximizes the profit of the extracted pieces (2). Constraint (3) 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 (4). The amount of extracted copies of some piece type must respect the demand for that piece type (a piece extracted is a piece sold) (5). Finally, the domain of all variables is the non-negative integers (6)–(7).

4.2 The revised variable enumeration

The variable enumeration described in [14] employs some rules to reduce the number of variables; they are symmetry-breaking, *Cut-Position*, and *Redundant-Cut*. The two last rules are not discussed here; [14] proves their correctness and they do not conflict with the enhanced model.

The use of the x variables does not change from the original formulation to our revised formulation – however, the size of the enumerated set of variables changes. Our 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.

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.

4.3 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. In [14], only the perfect symmetries described below are removed. Our changes 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 fits the plate, and the plate cannot fit an extra piece (of any type).

The second change alone cannot affect the model correctness. 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 our revised formulation there will always exist an extraction variable in such case: if a piece and plate match perfectly, there is no space for any other piece, fulfilling our only criteria for the existence of extraction variables. Consequently, what needs to be proved is that:

Theorem 1 *Without changing the pieces obtained from a packing, we may replace any normal cut after the middle of a plate by a combination of piece extractions and cuts at the middle of a plate or before it.*

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 fit at least one piece.
   (b) Its second child cannot fit 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.

We believe to be self-evident that the union of item 1,item 2a,items 2(b)i to 2(b)iii is equal to the set of all normal cuts after the middle of a plate. We present an individual proof for each of these cases.

Item 1 – *The cut has a perfect symmetry.* If two distinct cuts have the same children (with the only difference being the first child of one cut is the second child of the other cut, and vice-versa), then the cuts are perfectly symmetric. 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 we may keep only the cut in the first half of the plate.

Item 2a – *Its second child can fit at least one piece.* Proposition 1 allows us to replace the second child by a size-normalized 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-normalized counterpart may only be the same size or smaller. So the size-normalized 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) have stayed the same size or grown (because the size-normalization of its sibling), which guarantee 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 fit an extra piece together with the piece the first child would pack, or cannot fit any extra pieces. If it cannot fit any extra pieces, this fulfills our 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 fit 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 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 fit 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.

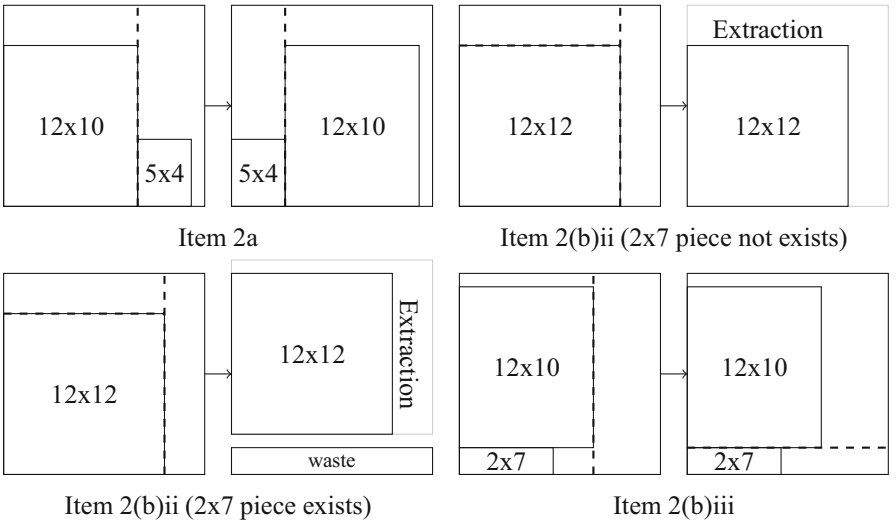

Fig. 3 Theorem 1 case examples. Item 1 and item 2(b)i are excluded given their simplicity. In all examples, the parent plate is 15x15. In the example of item 2a, the cut would happen after the middle of the plate, but then the pieces of the second child can be packed in the first child instead. In the example of item 2(b)i,, 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, we assume a 2x7 piece exists but we do not intend to obtain it from the plate; 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 happens first may be changed, as there is no piece packed in the subplate that would originally become the second child

Given the cases cover every cut after the middle of a plate, and each case has a proof, then follows that Theorem 1 is correct. □

5 The pricing phase

The pricing procedure described in [14, 29] was reimplemented by us. No significant changes were made to the procedure. As our experiments include multiple comparisons involving this procedure, a summary of the procedure is presented below. For simplicity, we consider the procedure takes an already built model (from either the original formulation or our enhanced version), and any previous reductions mentioned were already applied at this point.

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 (see [12, 14]).
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. Note 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. We 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 we used it in our experiments. In the last step, all variables yet fixed to zero are removed.

6 Experimental results

There are three formulation implementations that provide data used in our comparisons: *original* refers to the implementation presented in [14, 29]; *faithful* refers to our reimplementation of *original*; *enhanced* refers to our enhanced formulation presented

in Sect. 4. The *original* implementation was not available[3] Consequently, all data relative to *original* presented in this work comes from [29]. Both *faithful* and *enhanced* data were obtained by runs using the setup described in Sect. 6.1.

Each formulation may be modified by applying any combination of the following optional procedures: *priced* – refer to the pricing procedure described in [14, 29] 5 (same as [14, 29]); *normalized* – the plate-size normalization procedure described in Sect. 3; *warmed* – the MIP models solved were warm-started with a solution found by a previous step; *Cut-Position* and *Redundant-Cut* – are reduction procedures described in [14, 29], that may be enabled and disabled individually. For each experiment described in the next sections, if we do not mention a procedure, then it is disabled. The term *restricted priced* refers to the model for the position-only 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 a 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 [14]) 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 [14]).

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 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 now only be zero 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 Sect. 3. Hence, the pricing phase uncovers a set of unnecessary variables larger than the set it directly removes. Our preliminary experiments have shown that removing this larger set from the model, instead of just the variables directly removed by the pricing phase, has basically no effect on the total time; even if, on average, they account for 65% of the model variables after the pricing. Consequently, we believe the solver can detect such variables and remove them by itself. However, as it is computationally cheap to detect and remove the larger variable set, we decided to always apply this procedure after the pricing phase, and present a model size that better corresponds to reality in the results.

Each experiment helps to substantiate choices taken in the subsequent experiments: Sect. 6.2 provides evidence that *faithful* is on par with *original*, allowing us to use it as a replacement; Sect. 6.3 compares *faithful* to *enhanced* and shows the value of our contributions (namely, the *normalize* procedure and the *enhanced* formulation); Sect. 6.4 applies the methods with best results in the last experiment to prove new optimal values and bounds for harder instances.

[3] We asked the authors of [14] for the *original* implementation and Dimitri Thomopulos informed us it was not available.

6.1 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 task 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 [6] with JuMP 0.20.1 [13] and Gurobi 9.0.2 [15]. 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). For the root node relaxation of the final built model, the barrier algorithm was employed (`Method` $= 2$). Whenever the run included the pricing phase, the multiple continuous relaxations from such phase were solved by the dual simplex algorithm `Method` $= 1$. In preliminary experiments, barrier took less time than dual simplex to solve a model relaxation from scratch. However, if a previous base can be exploited, as it is the case during the pricing phase, choosing dual simplex over barrier made the pricing phase take less time.

6.2 Comparison of *faithful* against *original*

Without a reimplementation of *original*, any comparison would need to be made directly against the data in [29]. However, such 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 normalization* 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 1 compares the number of model variables and number of plates of the diverse model variants presented in [14, 29]. 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, all instances are either artificially generated, or of undisclosed origin. The number of enumerated plates has a strong correlation to the number of constraints in the model. Both [14] and [29] present the number of plates and not the number of constraints. To simplify the comparison, we do the same.

The *Priced PP-G2KP* runs in [14, 29] 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 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 position-only restricted model, or the iterated variable pricing, cannot be done in one hour, then the final model

Table 1 Comparison of *faithful* against *original*. 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 our 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

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
Priced PP-G2KP	3	7	14,619,460	31.92	1,642,382	25.55

will probably hit the time limit too – in [14], every run that hits one of the two first time limits also hits the third time limit. We chose to use a single three-hour time limit.

Table 1 references the names used in [14, 29]. 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 position-only restricted model.

The following conclusions can be derived from Table 1. All variants, except *Priced PP-G2KP*, are within $\pm 0.01\%$ of the expected number of plates (and, consequently, 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. Our reimplementation of *Redundant-cut* reduction seems responsible for both deviations. However, it follows closely the description given in [29]. 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 1 is sufficient. We cannot say the same for the *priced* variants. [14, 29] 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, we believe that the magnitude of the difference guarantees that we are not making a gross misrepresentation.

6.3 Comparison of *faithful* against *enhanced*

The primary purpose of this section is to evaluate our contributions to the state of the art. Our contributions are the *normalize* reduction (i.e., the plate-size normalization presented in Sect. 3) and the *enhanced* formulation (presented in Sect. 4.1). 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 [14, 29]. In this section, we use our reimplementation of *Complete PP-G2KP* named *faithful* (to distinguish from the data of the *original*). We also reimplemented the reductions and the pricing procedure, but as *enhanced* may also enable them, we avoid labelling these procedures as *faithful* as to avoid 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 a negligible amount of computational effort, and were already discussed in [14, 29].

We also examine the effects of 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 2 we can state that:

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. *normalize* further reduces variables by 14 $\sim$ 32% and plates by 37 $\sim$ 65%;
5. MIP-starting *enhanced* makes its 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. applying *pricing* to *faithful* is positive overall but loses one solved instance.

Considering the data from Table 3 we can state that:

1. both *E* and *H* phases are almost negligible (at most 2% with *H* in *enhanced*);
2. together the *RP* and *IP* phases account for 74.5 $\sim$ 83.5%;
3. *RP* and *IP* swap percentages between *enhanced* and *faithful*;
4. *faithful* shows some overhead in all phases strongly affected by model size.

Table 2 Comparison of *faithful* vs. *enhanced* over the 59 instances used in [29]. The meaning of the columns follow: *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. We count the time of these runs as they were timeouts

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

Table 3 Fraction of the total time spent in each step (only runs that executed all steps). *Time* is the sum of all time (in seconds) spent in the 47 instances that had all phases executed by all four variants considered. These are the same 47 indicated in row *Priced F. +N. +W.* of Table 2. 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

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

6.4 Evaluating *enhanced* against harder 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.

[30] proposes a set of 80 hard instances to test the limitations of their bounding procedures; we use these instances in this section. The instances were artificially generated and are divided in 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 vary 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

Table 4 Summary table for the instances proposed in [30]. The columns are: *C.* – instance class (described in [30], 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

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

dimension the original plate is larger; while, in the second class, the pieces are likely to 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, in average, the same dimension with double the size of the other; while, in the fourth class, half of the pieces follow the previous distribution, and the other half invert the favored dimension.

Only two configurations were selected for this experiment, the *priced* and non-*priced* versions of *enhanced* with *Cut-Position*, *normalize*, and *MIP-start* enabled. We also present the results for the *restricted priced* variant because it executes inside *priced* (the same reductions apply to it). Table 4 presents a summary of all runs, and Tables 5, 6, 7, and 8 present the improved bounds and solved instances.

For this experiment, Gurobi was allowed to use the 12 physical cores of our machine. Gurobi distributes the effort of the 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, we want 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 *position-only 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

Table 5 Instances solved (position-only restricted or unrestricted) or with improved bounds. Stopped with uselessly large upper bound. Value discarded to avoid breaking table formatting. We group lower and upper bounds that are valid for the unrestricted problem. 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 same instance and variant, if the LB and the UB are the same, 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 [30]

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

*Stopped with uselessly large upper bound. Value discarded to avoid breaking table formatting

time was shorter than non-*priced*, and 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 Tables 5, 6, 7, and 8 is to allow querying the exact values for specific instances. Even so, there are some gaps to fill. For the instances presented in Tables 5, 6, 7, and 8, 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, we can also summarize that our experiment has: proved 22

Table 6 See Table 5 for reference

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 7 See Table 5 for reference

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 8 See Table 5 for reference

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

unrestricted optimal values (5 already proven by [30], confirming their results); proved 22 restricted optimal values (in an overlapping but distinct subset of the instances); improved lower bounds for 25 instances; improved upper bounds for 58 instances.

7 Conclusions

The present work advances the state of the art on MILP formulations for the G2KP. We improve the performance of one of the most competitive MILP formulations for the G2KP by at least one order of magnitude. In the instance set selected by the original formulation, our enhanced formulation dominates the original formulation. Concerning other competitive MILP formulations in the literature, we keep the advantage of tighter bounds the original formulation had over them, and greatly reduce the model size and running times for instances that these other formulations had the advantage.

In the experiments, we have already discussed some elementary inferences, for example: the limitations (and partial success) of our improved formulation against the most recent and challenging instances in the literature; and the impact on the performance caused by the LP-solving algorithm, by the specific changes we made, by MIP-starting the models, and by some procedures proposed together with the original model (i.e., pricing and some preprocessing reductions). Here we present more general conclusions from a broader perspective.

We believe symmetry-breaking plays a significant part in the success of our enhanced formulation. In our experiments, we focus on the significant reduction of the model size because it is easier to measure. The solver can, however, always reduce the model size even further, by disregarding loose constraints or variables which cannot assume non-zero values. This does not seem to be the case of the variables removed

by our enhanced model, which could assume non-zero values and compose symmetric solutions. A single extraction variable may replace many distinct sequences of cuts that would extract the same piece from the same slightly-larger plate. The enhanced formulation did not present consistent gains in the LP relaxation, for them to be responsible for the observed improvement in performance. We also believe our 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 [14] or column generation techniques).

Our suggestions for future works follow: adapt the formulation for closely related problem variants and compare to their state-of-the-art solving procedure; expand on the symmetry-breaking; consider other frameworks besides the pricing framework of [14].

Acknowledgements We would like to thank Dimitri Thomopulos for informing us that the code used in [14, 29] was, unfortunately, not available anymore. Also, we would like to thank Mateus Pereira Martin for sharing his partial reimplementation of the code above mentioned, while not used in this paper his code allowed us to double-check our own reimplementation.

Author Contributions The authors' contributions may be summarised as: 1. an enhanced MILP formulation based on a previous state-of-the-art formulation; 2. the proof of correctness of the aforementioned enhancement; 3. empirical evidence of the better performance obtained by the enhancement; 4. the adaptation of a known reduction for our specific case; 5. empirical evidence of the gains obtained by this reduction in our case; 6.17 new optimal solution values of hard instances (unrestricted problem); 7. 22 new optimal solution values of hard instances (position-only restricted); 8. better lower bounds for 25 instances (unrestricted problem); 9. better upper bounds for 58 instances (unrestricted problem).

Funding This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)-Finance Code 001.

Data Availability The Jupyter notebooks and raw CSVs used for generating the tables in this work are available in .

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Code availability The code, in the specific version used, is available at [3]. The code is in public domain (by means of the Unlicense template, see https://unlicense.org/ for details).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Alvarez-Valdes, R., Parreño, F., Tamarit, J.M.: A branch and bound algorithm for the strip packing problem. OR Spectr. **31**(2), 431–459 (2009). https://doi.org/10.1007/s00291-008-0128-5
2. Beasley, J.E.: Algorithms for unconstrained Two-dimensional guillotine cutting. J. Oper. Res. Soc. **36**(4), 297–306 (1985). https://doi.org/10.1057/jors.1985.51.
3. Becker, H.: Guillotinemodels.jl (2022). https://doi.org/10.5281/zenodo.6486737. https://github.com/henriquebecker91/GuillotineModels.jl/tree/MPC_EXPERIMENTS
4. Belov, G.: Problems, models and algorithms in one- and two-dimensional cutting. Fakultät Mathematik und Naturwissenschaften der Technischen Universität Dresden (2003). https://d-nb.info/970782489/
5. Ben Messaoud, S., Chu, C., Espinouse, M.L.: Characterization and modelling of guillotine constraints. Eur. J. Oper. Res. **191**(1), 112–126 (2008). https://doi.org/10.1016/j.ejor.2007.08.029
6. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
7. Boschetti, M.A., Mingozzi, A., Hadjiconstantinou, E.: New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. IMA J. Manag. Math. **13**(2), 95–119 (2002). https://doi.org/10.1093/imaman/13.2.95
8. Christofides, N., Whitlock, C.: An algorithm for Two-Dimensional cutting problems. Oper. Res. **25**(1), 30–44 (1977). https://doi.org/10.1287/opre.25.1.30
9. Clautiaux, F., Sadykov, R., Vanderbeck, F., Viaud, Q.: Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. EURO J. Comput. Optim. **7**(3), 265–297 (2019). https://doi.org/10.1007/s13675-019-00113-9
10. Côté, J.F., Iori, M.: The meet-in-the-middle principle for cutting and packing problems. INFORMS J. Comput. **30**(4), 646–661 (2018). https://doi.org/10.1287/ijoc.2018.0806
11. Delorme, M., Iori, M.: Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. INFORMS J. Comput. **32**(1), 101–119 (2019). https://doi.org/10.1287/ijoc.2018.0880
12. Dolatabadi, M., Lodi, A., Monaci, M.: Exact algorithms for the two-dimensional guillotine knapsack. Computers Oper. Res. **39**(1), 48–53 (2012). https://doi.org/10.1016/j.cor.2010.12.018
13. Dunning, I., Huchette, J., Lubin, M.: Jump: a modeling language for mathematical optimization. SIAM Rev. **59**(2), 295–320 (2017). https://doi.org/10.1137/15M1020575
14. Furini, F., Malaguti, E., Thomopulos, D.: Modeling Two-Dimensional guillotine cutting problems via integer programming. INFORMS J. Comput. **28**(4), 736–751 (2016). https://doi.org/10.1287/ijoc.2016.0710
15. Gurobi Optimization, L.: Gurobi optimizer reference manual (2020). http://www.gurobi.com
16. Herz, J.C.: Recursive computational procedure for two-dimensional stock cutting. IBM J. Res. Dev. **16**(5), 462–469 (1972). https://doi.org/10.1147/rd.165.0462
17. Iori, M., de Lima, V.L., Martello, S., Miyazawa, F.K., Monaci, M.: Exact solution techniques for Two-dimensional cutting and packing. Eur. J. Oper. Res. p. S0377221720306111 (2020). https://doi.org/10.1016/j.ejor.2020.06.050. http://arxiv.org/abs/2004.12619. ArXiv: 2004.12619
18. Lodi, A., Monaci, M.: Integer linear programming models for 2-staged two-dimensional Knapsack problems. Math. Program. **94**(2), 257–278 (2003). https://doi.org/10.1007/s10107-002-0319-9
19. Martin, M., Birgin, E.G., Lobato, R.D., Morabito, R., Munari, P.: Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. Int. Transactions Oper. Res. **27**(2), 767–793 (2020). https://doi.org/10.1111/itor.12703
20. Martin, M., Morabito, R., Munari, P.: A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. Computers Oper. Res. **115** (2020). https://doi.org/10.1016/j.cor.2019.104851. http://www.sciencedirect.com/science/article/pii/S030505481930293X
21. Martin, M., Morabito, R., Munari, P.: A top-down cutting approach for modeling the constrained two- and three-dimensional guillotine cutting problems. J. Oper. Res. Soc. (2020). https://doi.org/10.1080/01605682.2020.1813640
22. Morabito, R., Belluzzo, L.: Optimising the cutting of wood fibre plates in the hardboard industry. Eur. J. Oper. Res. **183**(3), 1405–1420 (2007). https://doi.org/10.1016/j.ejor.2005.11.066
23. Parreño, F., Alonso, M.T., Alvarez-Valdes, R.: Solving a large cutting problem in the glass manufacturing industry. Eur. J. Oper. Res. **287**(1), 378–388 (2020). https://doi.org/10.1016/j.ejor.2020.05.016

24. Pisinger, D., Sigurd, M.: Using decomposition techniques and constraint programming for solving the Two-Dimensional bin-packing problem. INFORMS J. Comput. **19**(1), 36–51 (2007). https://doi.org/10.1287/ijoc.1060.0181
25. Russo, M., Boccia, M., Sforza, A., Sterle, C.: Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. Int. Transactions Oper. Res. **27**(2), 794–834 (2020). https://doi.org/10.1111/itor.12687
26. Scheithauer, G., Terno, J.: The g4-heuristic for the pallet loading problem. J. Oper. Res. Soc. **47**(4), 511–522 (1996). https://doi.org/10.1057/jors.1996.57
27. Song, X., Chu, C.B., Lewis, R., Nie, Y.Y., Thompson, J.: A worst case analysis of a dynamic programming-based heuristic algorithm for 2D unconstrained guillotine cutting. Eur. J. Oper. Res. **202**(2), 368–378 (2010). https://doi.org/10.1016/j.ejor.2009.05.047
28. Terno, J., Lindemann, R., Scheithauer, G.: Zuschnittprobleme und ihre praktische lösung. Verlag Harri Deutsch, Thun und Frankfurt/Main (1987)
29. Thomopulos, D.: Models and solutions of resource allocation problems based on integer linear and nonlinear programming. Ph.D. thesis, University of Bologna (2016). http://amsdottorato.unibo.it/7399/
30. Velasco, A.S., Uchoa, E.: Improved state space relaxation for constrained two-dimensional guillotine cutting problems. Eur. J. Oper. Res. **272**(1), 106–120 (2019). https://doi.org/10.1016/j.ejor.2018.06.016
31. Yanasse, H.H., Morabito, R.: A note on linear models for two-group and three-group two-dimensional guillotine cutting problems. Int. J. Prod. Res. **46**(21), 6189–6206 (2008). https://doi.org/10.1080/00207540601011543

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.