Packing by Scheduling: Using Constraint Programming to Solve a Complex 2D Cutting Stock Problem

Yiqing L. Luo and J. Christopher Beck

Department of Mechanical and Industrial Engineering
University of Toronto, Toronto, Ontario M5S 3G8, Canada
{louisluo, jcb}@mie.utoronto.ca

Abstract. We investigate the novel Two-stage Cutting Stock Problem with Flexible Length and Flexible Demand (2SCSP-FF): orders for rectangular items must be cut from rectangular stocks using guillotine cuts with the objective to minimize waste. Motivated by our industrial partner and different from problems in the literature, the 2SCSP-FF allows both the length of individual items and the total area of orders to vary within customer-specified intervals. We develop constraint programming (CP) and mixed-integer programming models, with the most successful coming from the adaptation of CP scheduling techniques. Numerical results show that this CP model has orders of magnitude smaller memory requirements and is the only model-based approach investigated that can solve industrial instances.

Keywords: Cutting Stock Problem · Guillotine Cuts · Constraint Programming · Mixed-Integer Linear Programming · Optimization.

1 Introduction

As scheduling is one of the most successful application areas of Constraint Programming (CP) [17, 19], we are interested in investigating whether CP scheduling approaches can be adapted to other combinatorial problems that share similar substructure. In this paper, we explore this idea for a complex, novel packing problem from the rolled-metal industry: the Two-stage Cutting Stock Problem with Flexible Length and Flexible Demand (2SCSP-FF), a generalization of the classic Two-stage Two-Dimensional Cutting Stock Problem with Guillotine Constraints (2SCSP). While the literature on 2SCSP is rich, our problem considers flexibility in item dimensions and total fulfillment, two characteristics that are commonplace in this industry [34], but have received limited attention in the literature. We approach these complications from a scheduling perspective, drawing inspiration from batch scheduling and the single resource transformation, a recently proposed CP modelling technique to handle the choice of alternative resources [3]. We conduct experiments over both generated and real-life

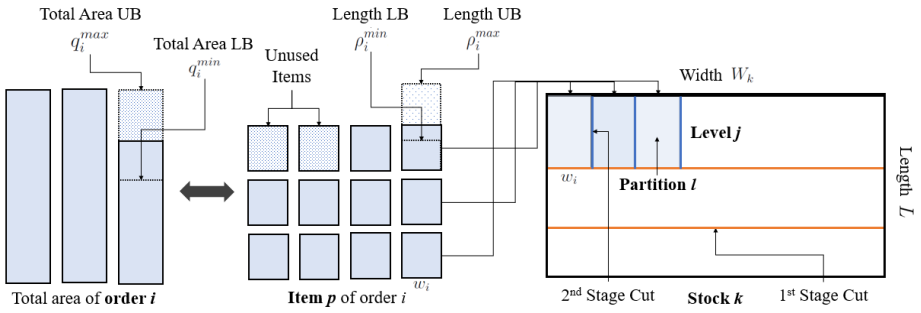

Fig. 1. A visualization of 2SCSP-FF. Each order is represented by its total quantity (left) and the partitions assigned to it (middle), as illustrated by the double-arrow. Dashed lines and the dotted fills indicate flexibility in the associated parameter. Orange and blue lines represent the first and second stage cuts, respectively.

instances and demonstrate our approach's computational advantages over alternative modelling approaches in CP, mixed integer programming (MIP), and a custom greedy heuristic.

Our contributions are as follows:

1. We introduce the novel 2SCSP-FF problem.
2. We adapt the single resource CP model to the 2SCSP-FF. This compact formulation increases the size of the instances that can be solved within memory and time limits by an order of magnitude.
3. We propose and experiment with alternative MIP and CP models and a two-stage heuristic.

2 Problem Definition

The 2SCSP-FF (Figure 1) is a novel generalization of the Two-stage Two-Dimensional Cutting Stock Problem with Guillotine Constraints (2SCSP). Given a set of *orders* for rectangular *items* and a set of larger *stock rectangles*, the classic Two-Dimensional Cutting Stock Problem (2DCSP) fulfills orders by cutting items from stocks. A more constrained variant, the 2SCSP only allows stocks to be processed using *guillotine cuts*, a cut that runs from one edge of the object to another. All cuts must also be executed in two stages, each consisting of a set of parallel guillotine cuts performed on a rectangle obtained from the previous stage. Without loss of generality, we let the direction of the first stage cuts be widthwise and that of the second stage ones lengthwise. The rectangles produced in the first stage are referred to as *levels*, following the literature [7], and those produced in the second stage as *partitions*. On top of 2SCSP, the 2SCSP-FF has the following characteristics arising from our application:

– **Flexible Length**: The length of an item is flexible within some integer interval. If a level contains items from different orders, its length must lie in

the intersection of the item-length intervals. In our application, items are subsequently rolled into cylindrical coils used as feedstock for downstream processing. The maximum length requirement ensures a maximum coil diameter to enable mounting it on a downstream machine. The minimum length requirement comes from the desire to limit the number of coils.

- **Flexible Demand**: Consistent with real-world manufacturing practices, each order can tolerate a percentage deviation from the total area demanded. For example, an order may request items totalling $10000 \pm 15\%$ units of area.
- **Maximum Partition Count per Level**: To reflect the limitations of an industrial cutter, the maximum number of partitions on each level is fixed.
- **Limited Stocks with Variable Widths**: Stock rectangles of various widths are available in limited quantities.
- **Cost Minimization**: The goal is to minimize cost: a weighted difference between the area of the stocks used and the area of the orders fulfilled.

Formally, we are given a set of stock rectangles K, whose types are characterized by set H. Each rectangle $k \in K$ has width W_k and length L. Stock rectangles with identical dimensions belong to the same type, K_h, $K = \bigcup_{h \in H} K_h$. We are also given a set of orders, N, where each order $i \in N$ has a required area interval of $[q_i^{min}, q_i^{max}]$. Each item belonging to order i should have a fixed width w_i and a length within the interval $[\rho_i^{min}, \rho_i^{max}]$. Due to the flexible length, the total number of items belonging to order i must be within an integer interval $[n_i^{min}, n_i^{max}] = [\lceil \frac{q_i^{min}}{\rho_i^{max}} \rceil, \lfloor \frac{q_i^{max}}{\rho_i^{min}} \rfloor]$. For order i, we denote its set of necessary items as $A_i = \{1, \ldots, n_i^{min}\}$, its set of possible, but not necessary items as $B_i = \{n_i^{min} + 1, \ldots, n_i^{max}\}$, and all possible items as $C_i = A_i \bigcup B_i$. Lastly, we let α and β be the weights associated with the area of stocks used and the area of orders fulfilled, respectively, and seek to minimize this weighted difference.

Since all stocks share the same length, a stock k can take on at most $\bar{j} = \lfloor \frac{L}{\min_{i \in N} \rho_i^{min}} \rfloor$ levels; we denote the set of possible numbers of levels of any stock as $J = \{0, \ldots, \bar{j}\}$. There must also be no more than η partitions on each level. We let $P = \{1, \ldots, \eta\}$ be the set of partitions on a given level, and $P_i = \{l \in P \mid l \leq n_i^{max}\}$ be the set of partitions on a given level assuming they are all assigned to order i. A partition of a stock that is assigned to an order becomes an item.

The 2SCSP-FF can easily be reduced to the 2SCSP if the quantity demanded of each order and the length of each item are fixed. As the 2SCSP is NP-hard [7], the 2SCSP-FF problem is at least NP-hard.

3 Literature Review

The 2SCSP was formalized by Gilmore and Gomory [9], who proposed a dynamic program and the well-known exponential-sized model solved via column generation. Since then, MIP has been the dominant approach in model-based studies. Limited to a single stock rectangle, Lodi and Monaci [22] proposed a compact formulation that restricts levelwise assignments for each item to reduce symmetry. Silva et al. [29] proposed a pseudo-polynomial formulation based on

the idea of cuts and residual plates for a 2SCSP with identical stocks. Furini et al. [7] extended both of these models to include different stock sizes while also proposing a branch-and-price algorithm based on Gilmore and Gomory's model. Macedo et al. [24] developed an arc flow formulation based on item positioning.

Dincbas and Simonis proposed the first CP-based approach [6] to the 2SCSP, generating stock patterns using a combination of backtrack search and a finite domain model. Later, Beldiceanu and Contejean introduced DIFFN [1, 2], a global constraint with an option to enforce guillotine cuts; however, no experimental results related to guillotine cuts were provided. Since then, CP has largely been investigated in other packing contexts. For the two-dimensional optimal rectangle packing problem, Korf [15, 16] considered solving a constraint satisfaction problem using the absolute positions of items. Moffitt and Pollack [26] studied the same satisfaction problem from a relative placement perspective, focusing on the pairwise relationships between items. For the same problem, Clautiaux et al. [5] considered a scheduling approach, representing the width and length of items as two interval variables. This was improved by Mesyagutov et al. [25], who integrated linear-programming-based pruning rules to propagate the constraints. Simonis and O'Sullivan investigated CP search strategies to pack squares into rectangles using the CUMULATIVE global constraint [30, 31]. For 1D packing, Shaw [28] proposed a global constraint PACK. For a comprehensive review of 2D packing problems, we refer the reader to surveys by Lodi et al. [21], Wäscher et al. [33] and Iori et al. [12].

While the 2SCSP has been widely studied, we could find only one work addressing item flexibility in the 2D setting. Lee et al. [20] considered a variant of the 2SCSP with flexible width and length and proposed a multi-stage heuristic to iteratively pack items and adjust level dimensions. They also proposed a non-linear model but did not investigate its performance.

CP techniques have been widely adopted in scheduling [17, 19]. In relation to our main approach, we discuss the literature on two types of problem: batch scheduling and vehicle routing. Batch scheduling arises when a set of jobs with common characteristics need to be processed together. Tang and Beck [32] proposed a CP formulation using interval variables for a multi-stage tool layup line problem. Ham and Cakici [10] used interval variables and state functions to represent a flexible job shop scheduling problem with parallel batch processing machines. Vehicle Routing Problems (VRP) optimize the routes of vehicles while some criteria associated with each route are satisfied. A number of CP models for VRP [4, 8, 13] represent the problem from a scheduling perspective with the trip-to-vehicle assignments modelled with some form of the ALTERNATIVE constraint. Recently, Booth and Beck [3] introduced the single resource model, where multiple resources are unified into a single resource on an expanded time horizon. They show that their formulation yields computation advantage over traditional modelling constructs in a capacity- and time-constrained routing problem.

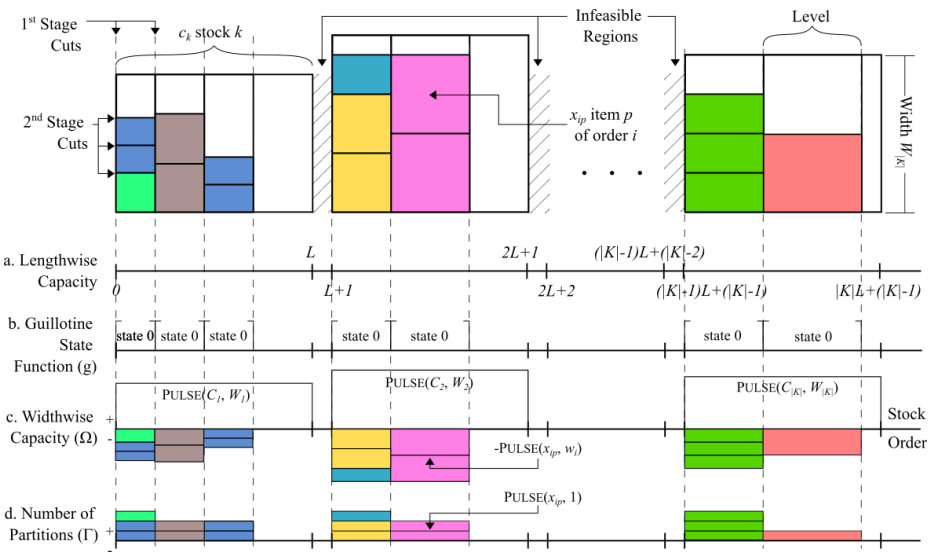

Fig. 2. Illustration of the CP_{SR} model. The length of the stock rectangles are concatenated along the horizontal axis. Here, a level is a vertical strip. The smaller rectangles and the dashed lines represent items and guillotine cuts, respectively.

4 The Single Resource CP Formulation

In this section, we present our main contribution: the Single Resource CP model, CP_{SR}. Our model poses the 2SCSP-FF as a scheduling problem composed of three main components: a unified domain of stock length, a state function for guillotine cuts, and cumulative functions tracking widthwise resources.

Unified Lengthwise Domain Our model adapts the single resource transformation [3], a CP modelling technique that unifies alternative resources into a single horizon, to the 2SCSP-FF. CP_{SR} concatenates the stock rectangles so that the total length of the stocks is analogous to a temporal horizon on which items belonging to all orders need to be allocated (Figure 2a). For each possible item $p \in C_i$ belonging to order i, we introduce an *optional interval variable* x_{ip}. In CP scheduling, an optional interval variable is a variable whose domain is a subset of $\{\perp\} \bigcup \{[s, \epsilon) | s, \epsilon \in \mathbb{Z}, s \leq \epsilon\}$, where s and ϵ are the start and end times of the interval, and $\perp$ is a special value indicating absence. In our case, the start time of x_{ip} represents an item's leftmost lengthwise coordinate, and the duration of x_{ip} its length, which we further restrict to be within $[\rho_i^{min}, \rho_i^{max}]$. For necessary items (i.e., in set A_i), we remove $\{\perp\}$ from the domain of x_{ip} for all $p \in A_i$ and simply declare them as *interval variables*.

To avoid an item spanning multiple stocks in the unified horizon, we insert a dummy unit of forbidden space between adjacent stocks to create an infeasible region (Figure 2a, hatched). The horizon is thus augmented from $L|K|$ to $(L +$

$1)|K|$, and no items can be placed in the infeasible region $\bar{F} = \bigcup_{k \in K}[Lk, (L + 1)(k)]$. We denote the augmented horizon as $\mathcal{H} = [0, L|K| + (|K| - 1)]$ and use ForbidExtent to exclude $\bar{F}$ from the domain of the item variables x_{ip}.

Guillotine State Function We draw inspiration from batch scheduling to model guillotine cuts: we treat each level in a stock rectangle as a batch so that the level's lengthwise endpoints coincide with the corresponding endpoints of the items. We first introduce a *state function*, g, a variable whose domain is a set of non-overlapping intervals (Figure 2b). Then, we associate the items with a level using an AlwaysConstant constraint, which coerces their interval variables to align with an interval in g. Thus, items can only be on the same level if they belong to the same interval in the state function.

Cumulative Resource Function Expressions The width of stocks and a level's partition count limit are interpreted as widthwise resources. Typical of CP scheduling, we use *cumulative functions* and *pulses*: the former are expressions that represent the sum of individual contributions of intervals over time, while the latter are expressions that indicate each interval's contribution. We let Ω, a cumulative function, be the net widthwise capacity over the horizon. In Ω, we generate a pulse with magnitude W_k for each stock rectangle k and a pulse with magnitude $-w_i$ for every item belonging to order i (Figure 2c). As long as Ω is non-negative, the widthwise capacity is satisfied. A similar construct is used to express the limit on the number of partitions on each level (Figure 2d), where a positive pulse with unit magnitude is generated for each item. We constrain the total cumulative function of these unit pulses, Γ, to be within η.

Overall, our decision variables are as follows:

- $x_{ip} \coloneqq$ (interval) lengthwise interval of item p belonging to order i.
- $c_k \coloneqq$ (interval) lengthwise interval representing stock k.
- $g \coloneqq$ (state function) guillotine state function.

CP_{SR} is defined in Model 1, using the syntax of IBM's CP Optimizer [11]. Objective (1a) defines our cost, the weighted difference between the areas of stocks used and orders fulfilled. Expressions PresenceOf and SizeOf are used to access the presence and the duration of an interval variable. If the variable is not present, both expressions evaluate to 0. Constraints (1b) and (1c) define the widthwise usage of each stock. The last two parameters in the AlwaysIn constraint respectively dictate the minimum and maximum values that the cumulative function Ω can take on over the horizon $\mathcal{H}$. Constraints (1d) and (1e) define the restriction on the number of partitions on each level. Constraint (1f) defines the guillotine cut restrictions. The last two parameters in AlwaysConstant ensure that the start and end times of the variables x_{ip} are aligned with those of the intervals within the state function g. Constraints (1g) and (1h) ensure that the total quantity of the order fulfilled is within the demand tolerance. Constraint (1i) ensures that no partition is assigned across two stock rectangles. The remaining constraints declare the decision variables.

$$\min\ \alpha \sum_{k\in K} LW_k \textsc{PresenceOf}(c_k) \qquad (CP_{SR}) \qquad (1a)$$

$$-\beta \sum_{i\in N}\sum_{p\in C_i} w_i \textsc{PresenceOf}(x_{ip})\textsc{SizeOf}(x_{ip})$$

$$\text{s.t.}\ \ \Omega = \sum_{k\in K} \textsc{Pulse}(c_k, W_k) - \sum_{i\in N}\sum_{p\in C_i} \textsc{Pulse}(x_{ip}, w_i) \qquad (1b)$$

$$\textsc{AlwaysIn}(\Omega, \mathcal{H}, 0, \max_{k\in K} W_k) \qquad (1c)$$

$$\Gamma = \sum_{i\in N}\sum_{p\in C_i} \textsc{Pulse}(x_{ip}, 1) \qquad (1d)$$

$$\textsc{AlwaysIn}(\Gamma, \mathcal{H}, 0, \eta) \qquad (1e)$$

$$\textsc{AlwaysConstant}(g, x_{ip}, True, True) \qquad \forall i \in N, p \in C_i \quad (1f)$$

$$\sum_{p\in C_i} \textsc{SizeOf}(x_{ip}) \geq q_i^{min}/w_i \qquad \forall i \in N \qquad (1g)$$

$$\sum_{p\in C_i} \textsc{SizeOf}(x_{ip}) \leq q_i^{max}/w_i \qquad \forall i \in N \qquad (1h)$$

$$\textsc{ForbidExtent}(x_{ip}, \bar{F}) \qquad \forall i \in N, p \in C_i \quad (1i)$$

$$x_{ip} : \textsc{IntervalVar}(\mathcal{H}, [\rho_i^{min}, \rho_i^{max}]) \qquad \forall i \in N, p \in A_i \quad (1j)$$

$$x_{ip} : \textsc{OptIntervalVar}(\mathcal{H}, [\rho_i^{min}, \rho_i^{max}]) \qquad \forall i \in N, p \in B_i \quad (1k)$$

$$c_k : \textsc{IntervalVar}([(L+1)(k-1), (L+1)k-1], L) \forall k \in K \qquad (1l)$$

$$g : \textsc{StateFunction}() \qquad (1m)$$

5 Alternative Approaches

We also propose integer-based CP and MIP models and a two-stage heuristic.

5.1 Integer-based CP Formulations

Counting-based CP Model Due to the two-stage cuts, partitions assigned to the same order on a level must be identical; hence, we can count them. For a given level j from stock k, we use an integer variable x_{ijk} to denote the number of partitions assigned to order i. We use an integer variable y_{jk} to represent the length of level j on stock k. As the position of the lengthwise cut is not restricted to be integral, we magnify the domain using a precision parameter $\mathcal{P}$ equal to some power of ten, so that y_{jk} represents the first $\log_{10}\mathcal{P}$ decimal places of the actual length. More formally, our decision variables are as follows:

- $x_{ijk} \coloneqq$ (integer) # of partitions on level j of stock k assigned to order i
- $y_{jk} \coloneqq$ (integer) length of level j of stock k magnified by $\mathcal{P}$

$$\min \ \alpha \sum_{k\in K} LW_k c_k - \beta \sum_{i\in N}\sum_{j\in J}\sum_{k\in K} w_i x_{ijk} y_{jk}/\mathcal{P} \quad (CP_{CO}) \tag{2a}$$

$$\text{s.t.} \ \sum_{i\in N} x_{ijk} w_i \le W_k s_{jk} \quad \forall j \in J, k \in K \tag{2b}$$

$$y_{jk}/\mathcal{P} \le \rho_i^{max} + (x_{ijk} == 0) \max_{n\in N}(\rho_n^{max}) \quad \forall i \in N, j \in J, k \in K \tag{2c}$$

$$y_{jk}/\mathcal{P} \ge \rho_i^{min}(x_{ijk} \ge 1) \quad \forall i \in N, j \in J, k \in K \tag{2d}$$

$$\sum_{j\in J}\sum_{k\in K} y_{jk} x_{ijk}/\mathcal{P} \ge q_i^{min}/w_i \quad \forall i \in N \tag{2e}$$

$$\sum_{j\in J}\sum_{k\in K} y_{jk} x_{ijk}/\mathcal{P} \le q_i^{max}/w_i \quad \forall i \in N \tag{2f}$$

$$\sum_{j\in J} y_{jk}/\mathcal{P} \le L c_k \quad \forall k \in K \tag{2g}$$

$$s_{jk} = \textsc{Any}([x_{ijk} > 0, \forall i \in N]) \quad \forall j \in J, k \in K \tag{2h}$$

$$c_k = \textsc{Any}([s_{jk} = 1, \forall j \in J]) \quad \forall k \in M \tag{2i}$$

$$x_{ijk} \in \{0, ..., \eta\} \quad \forall i \in N, j \in J, k \in K \tag{2j}$$

$$y_{jk} \in \{0, \min_{i\in N} \rho_i^{min}\mathcal{P}, \ldots, \max_{i\in N} \rho_i^{max}\mathcal{P}\} \quad \forall j \in J, k \in K \tag{2k}$$

Model 2 formalizes CP_{CO}. Objective (2a) describes the cost. Since y_{jk} is magnified, we divide it by $\mathcal{P}$ to recover its actual length. Constraint (2b) restricts the width of the stocks. Constraints (2c) and (2d) restrict the length of a level by the tightest interval determined by the allotted orders. Constraints (2e) and (2f) ensure that partitions of each order fulfilled satisfy the total quantity range demanded. Constraint (2g) restricts the length of the stocks in use. Constraints (2h) and (2i) describe if a level and a stock is used, respectively.

Stock-based CP Model Extending the standard integer-based CP model for one-dimensional bin packing [14], the stock-based CP model, CP_{ST}, takes advantage of the limited number of possible partitions on a level, matching each partition to some order. Specifically, we define integer variables x_{jkl} representing the index of the order to which the l^{th} partition of the j^{th} level on the k^{th} stock is assigned. As not all partitions are always needed, we define a dummy order that serves as a placeholder. Formally, the dummy order, indexed by $D = |N| + 1$, has width $w_D = 0$ and length interval $[\rho_D^{min}, \rho_D^{max}] = [0, \max_{i\in N}(\rho_i^{max})]$. We use $\overline{N} = N \bigcup \{D\}$ to denote the set of original orders plus the dummy order; $\overline{w}$ to denote the set of widths of original orders union the dummy width w_D; $\overline{\rho^{min}}$ and $\overline{\rho^{max}}$ to denote the lengthwise bounds of orders union the dummy bounds ρ_D^{min} and ρ_D^{max}. Similar to CP_{CO}, we let y_{jk} be the length of level j on stock k and magnify its domain using $\mathcal{P}$.

$$\text{minimize} \quad \sum_{k\in K} LW_k c_k - \sum_{j\in J}\sum_{k\in K}\sum_{l\in P} \overline{w}_{x_{jkl}} y_{jk}/\mathcal{P} \qquad (CP_{ST}) \qquad (3a)$$

$$\text{s.t.} \quad \sum_{l\in P} \overline{w}_{x_{jkl}} \leq W_k s_{jk} \qquad \forall j \in J, k \in K \qquad (3b)$$

$$y_{jk}/\mathcal{P} \leq \overline{\rho^{max}}_{x_{jkl}} \qquad \forall j \in J, k \in K, l \in P \qquad (3c)$$

$$y_{jk}/\mathcal{P} \geq \overline{\rho^{min}}_{x_{jkl}} \qquad \forall j \in J, k \in K, l \in P \qquad (3d)$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{l\in P} (x_{jkl} == i) y_{jk}/\mathcal{P} \geq q_i^{min}/w_i \qquad \forall i \in N \qquad (3e)$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{l\in P} (x_{jkl} == i) y_{jk}/\mathcal{P} \leq q_i^{max}/w_i \qquad \forall i \in N \qquad (3f)$$

$$s_{jk} = \textsc{Any}([x_{jkl} \neq D, \forall l \in P]) \qquad \forall j \in J, k \in K \qquad (3g)$$

$$x_{jkl} \in \overline{N} \qquad \forall j \in J, k \in K, l \in P \qquad (3h)$$

$$(2g), (2i), (2k)$$

Model 3 formalizes CP_{ST}. Objective (3a) minimizes the cost. Constraint (3b) ensures that the widthwise capacity is satisfied on each stock. In particular, $\overline{w}$ is indexed by x_{jkl} using the Element constraint. Constraints (3c) and (3d) constrain the length of a level by the items assigned on it. Constraints (3e) and (3f) satisfy the total area of each order. Constraint (3g) instantiates intermediate parameters indicating level usage.

Modelling Considerations

Symmetry-breaking: The problem has a number of inherent symmetries due to the homogenous items, levels, and stock rectangles. Hence, we augment CP_{CO} and CP_{ST} with the following symmetry-breaking constraints:

$$y_{jk} \geq y_{(j+1)k} \qquad \forall j \in J', k \in K \qquad (4a)$$

$$c_k \geq c_{k+1} \qquad \forall k \in K'_h, h \in H \qquad (4b)$$

These constraints break the symmetry between the lengths of consecutive levels on the same stock and the presence of homogeneous stocks, respectively. We use a prime to indicate an ordered set without its last element: $J' = J \setminus \{|J|\}$. For CP_{ST}, we also specify a lexicographic ordering of the order indices on consecutive levels of the same stock via $\textsc{Lexicographic}([x_{jkl}, \forall l \in P], [x_{(j+1)kl}, \forall l \in P])$.

Item-based CP Model: Using Pack [28], we can also construct an integer-based CP model that decides on the level that an item is assigned to. Two such structures exist in 2SCSP-FF: the packing of items into levels and that of levels into stocks. While the former can be represented by Pack, the latter cannot due to the lengthwise flexibility and is represented by constraints that decompose Pack. The model is omitted due to poor computational performance.

5.2 Mixed-Integer Formulation

We also introduce a mixed-integer program, *MIP*, that uses binary variables to assign partitions on each level to orders. Formulating a strong MIP model is challenging, as determining the area of each order requires information related to two independent decisions: the order-to-level assignment and the level length given order assignments. In *MIP*, we linearize this relationship at the expense of introducing new variables, each one packing an item of an order into a level of a stock. While it is tempting to decompose them into independent orders-to-levels and levels-to-stocks decisions similar to the compact formulation in Furini et al. [7], representing both the area of each order and the variable length of each level using linear constraints is nontrivial. Here, we do not investigate this further.

More formally, our decision variables are as follows:

- $x_{ijkl} \coloneqq$ (binary) 1 if the l^{th} partition from the j^{th} level of the k^{th} stock is assigned to the i^{th} order, else 0.
- $y_{jk} \coloneqq$ (continuous) the length of the j^{th} level on the k^{th} stock.
- $a_{ijkl} \coloneqq$ (continuous) the area occupied by the l^{th} partition of the j^{th} level on the k^{th} stock belonging to order i.
- $c_k \coloneqq$ (binary) 1 if the k^{th} stock is used, else 0.

MIP is defined in Model 5. Objective (5a) describes the cost. Constraint (5b) restricts the stocks' width. Constraint (5c) limits the number of lengthwise cuts. Constraint (5d) ensures that the lengthwise capacity of each stock is satisfied. Constraints (5e) and (5f) assert that the level's length must respect the minimum and maximum length of items assigned to it. Constraints (5g) and (5h) ensure that the quantity of each order assigned across all stocks is satisfactory. Constraints (5i), (5j), and (5k) define the area of each partition on a level.

$$\min \ \alpha \sum_{k \in K} LW_k c_k - \beta \sum_{i \in N, j \in J, k \in K, l \in P_i} a_{ijkl} \qquad (MIP) \qquad (5a)$$

$$\text{s.t.} \sum_{i \in N, l \in P_i} w_i x_{ijkl} \leq W_k c_k \qquad \forall j \in J, k \in K \qquad (5b)$$

$$\sum_{i \in N, l \in P_i} x_{ijkl} \leq \eta c_k \qquad \forall j \in J, k \in K \qquad (5c)$$

$$\sum_{j \in J} y_{jk} \leq L c_k \qquad \forall k \in K \qquad (5d)$$

$$y_{jk} \geq \rho_i^{min} x_{ijkl} \qquad \forall i \in N, j \in J, k \in K, l \in P_i \qquad (5e)$$

$$y_{jk} \leq \rho_i^{max} x_{ijkl} + \max_{i' \in N}(\rho_{i'}^{max})(1 - x_{ijkl}) \qquad \forall i \in N, j \in J, k \in K, l \in P_i \qquad (5f)$$

$$\sum_{l \in P_i, j \in J, k \in K} a_{ijkl} \geq q_i^{min} \qquad \forall i \in N \qquad (5g)$$

$$\sum_{l \in P_i, j \in J, k \in K} a_{ijkl} \leq q_i^{max} \qquad \forall i \in N \qquad (5h)$$

$$a_{ijkl} \leq w_i y_{jk} \qquad \forall i \in N \qquad (5i)$$

$$a_{ijkl} \geq w_i y_{jk} - \rho_i^{max} w_i (1 - x_{ijkl}) \qquad \forall i \in N \qquad (5j)$$
$$a_{ijkl} \leq \rho_i^{max} w_i x_{ijkl} \qquad \forall i \in N, j \in J, k \in K, l \in P_i \qquad (5k)$$
$$x_{ijkl} \in \{0, 1\} \qquad \forall i \in N, j \in J, k \in K, l \in P_i \qquad (5l)$$
$$y_{jk} \in \mathbb{R}^+ \qquad \forall j \in J, k \in K \qquad (5m)$$
$$a_{ijkl} \in \mathbb{R}^+ \qquad \forall i \in N, j \in J, k \in K, l \in P_i \qquad (5n)$$
$$c_k \in \{0, 1\} \qquad \forall k \in K \qquad (5o)$$

Modelling Considerations

Symmetry-breaking: We can again add symmetry-breaking constraints (4a) and (4b) to *MIP* similar to CP_{CO} and CP_{ST}. Furthermore, we add constraints (6a) and (6b) to break the symmetry between partitions on the same level belonging to the same order and the length of the first level of identical stocks, respectively.

$$x_{ijkl} \geq x_{ijk(l+1)} \qquad \forall i \in N, j \in J, k \in K, l \in P_i' \qquad (6a)$$
$$y_{0k} \geq y_{0(k+1)} \qquad \forall k \in K_h', h \in H \qquad (6b)$$

One-hot Encoded Formulation: In order to retain linearity, *MIP* treats the assignment of different partitions on the same level to an order as individual decisions. Alternatively, we can one-hot encode the number of partitions assigned to the level so that each binary variable x_{ijkl} takes the value 1 if and only if there are l partitions (with identical dimensions) on level j of stock k assigned to order i. This formulation underperforms *MIP* in our experiments and is omitted.

5.3 First-fit Based Heuristic

In addition to the mathematical models, we develop a two-stage first-fit-based heuristic, *FFMH*. The first stage sorts the orders' items in a lexicographically decreasing order based on their width and length interval size and packs each one into a level. The intuition is that orders with less lengthwise flexibility and larger width should be packed into a level first, as they can be more difficult to pack into a partial solution. A new level or stock is opened if an item cannot fit into the previous level or stock. Packing an item into a stock's level only narrows its length interval: another decision is required to obtain its exact length and thereafter each order's total area. For simplicity, we pack items of an order until the sum of the average possible area of each item is not less than the middle of the required area interval for that order. In the second stage, given the complete item-to-level assignment, we solve a linear program (Model 7) to determine each level's length, while minimizing cost.[1]

[1] The form of this two-stage heuristic suggests that a classical Benders decomposition approach may be worth investigation in future work.

$$\max \sum_{i\in N, j\in J, k\in K} w_i \Phi_{ijk} y_{jk} \tag{7a}$$

$$\text{s.t.} \sum_{j\in J} y_{jk} \leq L \qquad \forall k \in K \tag{7b}$$

$$y_{jk} \geq \Phi^{ind}_{ijk} \rho^{min}_i \qquad \forall i \in N, j \in J, k \in K \tag{7c}$$

$$y_{jk} \leq \Phi^{ind}_{ijk} \rho^{max}_i + (1 - \Phi^{ind}_{ijk}) \max_{i'\in N}(\rho^{max}_{i'}) \forall i \in N, j \in J, k \in K \tag{7d}$$

$$\sum_{j\in J, k\in K} w_i \Phi_{ijk} y_{jk} \geq q^{min}_i \qquad \forall i \in N \tag{7e}$$

$$\sum_{j\in J, k\in K} w_i \Phi_{ijk} y_{jk} \leq q^{max}_i \qquad \forall i \in N \tag{7f}$$

$$y_{jk} \in \mathbb{R}^+ \qquad \forall j \in J, k \in K \tag{7g}$$

The only variables in Model 7 are the continuous variables y_{jk} describing the length of the j^{th} level on the k^{th} stock. The parameter Φ_{ijk} is the number of partitions on the j^{th} level of the k^{th} stock that belongs to order i, and the parameter Φ^{ind}_{ijk} is a 0-1 indicator for $\Phi_{ijk} > 0$. We simplify the cost minimization objective to maximize total fulfillment (7a) because the number of stocks used is fixed given the item-to-level assignment. Constraint (7b) constrains the stock length. Constraints (7c) and (7d) satisfy the length specifications of the partitions. Constraints (7e) and (7f) ensure the total fulfillment of each order to be within tolerance limits. Constraint (7g) declares the variable domain.

6 Numerical Results

We conduct our analysis on a combination of 50 generated problem instances and 4 real-life instances provided by our industry partner (Table 1).

For the generated instances, we draw from distributions provided by our industrial collaborator (Table 2), generating 10 instances for each parameter combination in the set $\{(|N|, |K|)\} \in \{(4, 8), (8, 16), (16, 32), (32, 64), (64, 128)\}$. For 5 out of these 10 instances, we halve the total area tolerances to provide variability. In the rare case that, for some order i, $\rho^{max}_i \leq \rho^{min}_i$ is generated, we swap the two values.

$\|N\|$	$\|K\|$	W_k	w_i	ρ^{min}_i	ρ^{max}_i	q^{min}_i	q^{max}_i
19	42	48	22.2	112.9	180.2	1.3e5	1.7e5
21	172	45.2	16.3	56.6	94.6	9.7e4	1.3e5
47	149	43.3	10.4	68.2	134.1	1.6e5	2.1e5
149	636	44.6	13.3	74.4	134.4	2.5e5	3.4e5

Table 1. Mean of the parameter combinations from the four industrial instances.

Parameter	Distribution
	$\forall$ order $i \in N$
q_i	Exponential(λ=5.608e-0.5)
q_i^{max}	Constant, 0.85 q_i
q_i^{min}	Constant, 1.15 q_i
w_i	Integer Uniform(a=1, b=20)
ρ_i^{max}	Integer Uniform(a=70, b=115)
ρ_i^{min}	Integer Uniform(a=85, b=130)
	$\forall$ stock $k \in K$
W_k	Integer Uniform(a=36, b=50) with 50% chance of duplicating previous stock
L	Constant, 400

Table 2. Data distributions for each parameter.

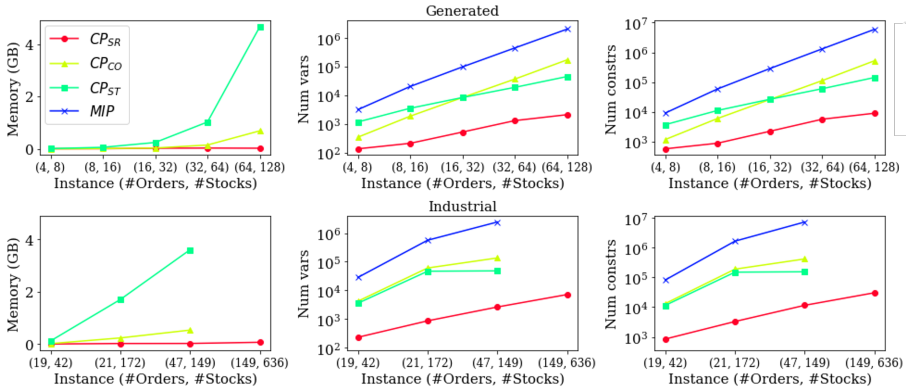

Fig. 3. Comparison of the number of variables, the number of constraints, and the model memory before search over instance sizes. Note the scales of the y-axes.

All experiments are implemented in Python 3.8, and computations were performed on individual nodes of the SciNet Niagara cluster [23, 27]. We use CPLEX and CP Optimizer from the CPLEX Optimization Studio version 20.1.0 via the DOcplex library. Each model is given 16 GB of RAM and runs that exceed this size are aborted. All experiments are single-threaded with default search and inference settings. A one-hour time limit is used.

For the CP integer-based models, we set $\mathcal{P}$, the magnifying parameter, to 1, as increasing it led to poor performance. We set α and β, the objective weights, to 0.3 and 0.7, respectively, to reflect the industrial use case.

Model Size Comparison. Figure 3 compares the mean model sizes based on the number of variables, the number of constraints, and the model memory (before search). The memory usage of *MIP* is not accessible from the solver. A data point is omitted if the corresponding model fails to initialize in memory within the one hour time limit. The CP_{SR} formulation is significantly smaller than the other models across all three measures, especially as the instances scale up. For

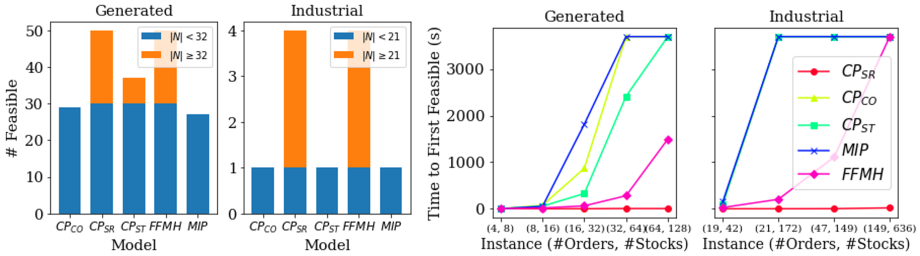

Fig. 4. (Left) Number of generated and industrial instances with a feasible solution by each model. (Right) The average run time required to find a feasible solution for a given model at an instance size.

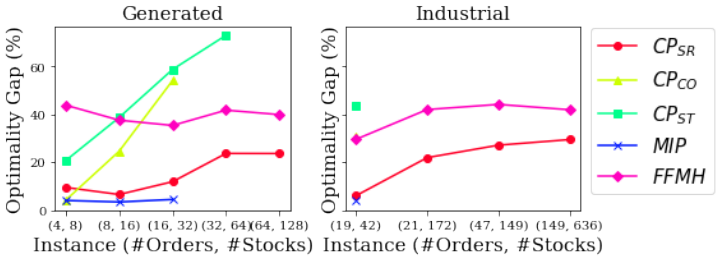

Fig. 5. % optimality gap for generated and industrial instances. Instances that are not solved by an approach are not included in that approach's measure.

the largest industrial instances, no other models could be loaded before timing out. For the largest generated instances, $(64, 128)$, the CP_{CO} model requires about 800MB of memory, while CP_{SR} only needs 20MB.

Feasibility Analysis. Figure 4 reports the number of instances for which a feasible solution was found and the average time to feasibility or termination. Only CP_{SR} and *FFMH* found a feasible solution to all instances. In particular, CP_{SR} reached feasibility the fastest amongst all methods, requiring less than 100 seconds for the largest industrial instance, while *FFMH*, the second fastest, needed almost the entire one-hour run time. Notably, *MIP* failed to find feasible solutions for generated and industrial instances with $|N| \geq 32$ and $|N| \geq 21$, respectively.

Solution Quality Figure 5 displays the optimality gap of each approach calculated from Equation (8). Here, $z(n; i)$ is the objective value of approach n for instance i, and $lb(i)$ is the best lower bound of instance i across all approaches. For a given solution approach, we omit any unsolved instances from the visualization; hence, CP_{SR} is penalized for finding solutions to harder instances compared to approaches that did not do so.

$$\%\ \text{OptGap}(n;\ i) = 100\frac{z(n;i) - lb(i)}{lb(i)} \tag{8}$$

MIP demonstrated the strongest performance for generated instances with $|N| \leq 16$. It proved optimality for three generated instances, the only ones proven optimal across all models. For larger instances ($|N| \geq 32$), *MIP* scaled poorly, failing to find a feasible solution within the time limit. Both CP_{CO} and CP_{ST} struggle to find competitive solutions to the generated instances past $N = 4$, eventually encountering loading time issues for larger instances. Notably, CP_{CO} found similar solutions to *MIP* for the smallest generated instances, but could not prove optimality due to a weaker lower bound. CP_{SR} consistently outperformed *MIP* and the other CP models for all but the smaller instances. A similar trend is observed on the industrial instances, where CP_{SR} was the only model-based approach that found a feasible solution to more than one instance. For both sets of instances, CP_{SR} consistently found better solutions than the heuristic in less time. We also observe that the optimality gaps of the larger instances that only CP_{SR} can solve are of the same order of magnitude as the gaps of the smaller ones. The lower bounds of these large instances are generated by CP_{SR}, but their values are non-trivial, a rare feat for typical CP approaches. We note that CP Optimizer computes the lower bound using an automatic LP-based relaxation of the scheduling constraints [18], a feature not available in some other CP solvers.

7 Discussion and Conclusion

In this paper, we create a CP scheduling approach for a novel packing problem: the Two-stage Cutting Stock Problem with Flexible Length and Flexible Demand (2SCSP-FF). Using optional intervals, state functions, and cumulative functions, our model, CP_{SR}, has significant computational and performance advantages over two alternative CP models, a MIP model, and a two-stage heuristic on large generated and industrial instances.

The memory efficiency of CP_{SR} can be attributed to the compact representation of the complicated substructures. To represent the guillotine cuts, CP_{SR} is the only model that does not enumerate over the set of levels J, instead using just a state function and a AlwaysConstant constraint. Similarly, CP_{SR} restricts the widthwise capacities and the partition counts without levelwise constraints. By using the fewest variables and constraints, the CP_{SR} model has at least an order-of-magnitude savings in its memory usage. As the instances scale up, this advantage increases.

Accordingly, only CP_{SR} found a feasible solution to more than one industrial-scale instance. The short time-to-feasibility, however, differs from the results for routing problems [3], where the model struggled to find feasible solutions quickly. We suspect that this disparity is due to the looser constraints on interval variables for our problem compared to the routing formulation.

Overall, our success here suggests that the flexibility of CP scheduling tools provides a promising approach to attacking complex real-world problems beyond traditional scheduling ones.

Acknowledgements. We thank anonymous reviewers for their valuable feedback. This research was partially supported by Visual Thinking International Ltd (Visual8) and the Natural Sciences and Engineering Research Council of Canada.

References

1. Beldiceanu, N., Carlsson, M., Flener, P., Pearson, J.: On the reification of global constraints. Constraints **18**, 1–6 (2012)
2. Beldiceanu, N., Contejean, E.: Introducing global constraints in chip. Mathematical and Computer Modelling **20**(12), 97–123 (1994)
3. Booth, K.E.C., Beck, J.C.: A constraint programming approach to electric vehicle routing with time windows. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019. Lecture Notes in Computer Science, vol. 11494, pp. 129–145. Springer (2019)
4. Cappart, Q., Schaus, P.: Rescheduling railway traffic on real time situations using time-interval variables. In: Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017. Lecture Notes in Computer Science, vol. 10335, pp. 312–327. Springer (2017)
5. Clautiaux, F., Jouglet, A., Carlier, J., Moukrim, A.: A new constraint programming approach for the orthogonal packing problem. Computers & Operations Research **35**(3), 944–959 (2008), part Special Issue: New Trends in Locational Analysis
6. Dincbas, M., Simonis, H., Hentenryck, P.V.: Solving a cutting-stock problem with the constraint logic programming language CHIP. Mathematical and Computer Modelling **16**, 95–105 (1992)
7. Furini, F., Malaguti, E.: Models for the two-dimensional two-stage cutting stock problem with multiple stock size. Computers & Operations Research **40**(8), 1953–1962 (2013)
8. Gedik, R., Kirac, E., Milburn, A.B., Rainwater, C.: A constraint programming approach for the team orienteering problem with time windows. Computers & Industrial Engineering **107**, 178–195 (2017)
9. Gilmore, P.C., Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. Operations Research **13**(1), 94–120 (1965)
10. Ham, A.M., Cakici, E.: Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. Computers & Industrial Engineering **102**, 160–165 (2016)
11. IBM: CP optimizer user manual, `https://www.ibm.com/docs/en/icos/20.1.0?topic=optimizer-cp-users-manual`
12. Iori, M., de Lima, V.L., Martello, S., Miyazawa, F.K., Monaci, M.: Exact solution techniques for two-dimensional cutting and packing. European Journal of Operational Research **289**(2), 399–415 (2021)
13. Kinable, J., van Hoeve, W., Smith, S.F.: Optimization models for a real-world snow plow routing problem. In: Integration of AI and OR Techniques in Constraint Programming - 13th International Conference, CPAIOR 2016. Lecture Notes in Computer Science, vol. 9676, pp. 229–245. Springer (2016)
14. Kong, V.L.: IBMDecisionOptimization: Docplex-Examples/Trimloss.py. `https://github.com/IBMDecisionOptimization/docplex-examples/blob/master/examples/cp/basic/trimloss.py` (2020)
15. Korf, R.E.: Optimal rectangle packing: Initial results. In: Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003). pp. 287–295. AAAI (2003)

16. Korf, R.E.: Optimal rectangle packing: New results. In: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004). pp. 142–149. AAAI (2004)
17. Ku, W., Beck, J.C.: Mixed integer programming models for job shop scheduling: A computational analysis. Computers & Operations Research **73**, 165–173 (2016)
18. Laborie, P., Rogerie, J.: Temporal linear relaxation in IBM ILOG CP optimizer. Journal of Scheduling **19**(4), 391–400 (2016)
19. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP optimizer for scheduling - 20+ years of scheduling with constraints at IBM/ILOG. Constraints **23**(2), 210–250 (2018)
20. Lee, J., Kim, B.I., Johnson, A.L.: A two-dimensional bin packing problem with size changeable items for the production of wind turbine flanges in the open die forging industry. IIE Transactions **45**, 1332 – 1344 (2013)
21. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research **141**(2), 241–252 (2002)
22. Lodi, A., Monaci, M.: Integer linear programming models for 2-staged two-dimensional knapsack problems. Mathematical Programming **94**(2-3), 257–278 (2003)
23. Loken, C., Gruner, D., Groer, L., Peltier, R., Bunn, N., Craig, M., Henriques, T., Dempsey, J., Yu, C.H., Chen, J., Dursi, L.J., Chong, J., Northrup, S., Pinto, J., Knecht, N., Zon, R.V.: SciNet: Lessons learned from building a power-efficient top-20 system and data centre. Journal of Physics: Conference Series **256**, 012026 (nov 2010)
24. Macedo, R., Alves, C., de Carvalho, J.M.V.: Arc-flow model for the two-dimensional guillotine cutting stock problem. Computers & Operations Research **37**(6), 991–1001 (2010)
25. Mesyagutov, M., Scheithauer, G., Belov, G.: LP bounds in various constraint programming approaches for orthogonal packing. Computers & Operations Research **39**(10), 2425–2438 (Oct 2012)
26. Moffitt, M.D., Pollack, M.E.: Optimal rectangle packing: A meta-csp approach. In: Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, (ICAPS 2006). pp. 93–102. AAAI (2006)
27. Ponce, M., van Zon, R., Northrup, S., Gruner, D., Chen, J., Ertinaz, F., Fedoseev, A., Groer, L., Mao, F., Mundim, B.C., Nolta, M., Pinto, J., Saldarriaga, M., Slavnic, V., Spence, E., Yu, C., Peltier, W.R.: Deploying a top-100 supercomputer for large parallel workloads: the niagara supercomputer. In: Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), PEARC 2019, Chicago, IL, USA, July 28 - August 01, 2019. pp. 34:1–34:8. ACM (2019)
28. Shaw, P.: A constraint for bin packing. In: Principles and Practice of Constraint Programming – CP 2004. pp. 648–662. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
29. Silva, E., Alvelos, F., Valério de Carvalho, J.: An integer programming model for two- and three-stage two-dimensional cutting stock problems. European Journal of Operational Research **205**(3), 699–708 (2010)
30. Simonis, H., O'Sullivan, B.: Search strategies for rectangle packing. In: Stuckey, P.J. (ed.) Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5202, pp. 52–66. Springer (2008), https://doi.org/10.1007/978-3-540-85958-1_4

31. Simonis, H., O'Sullivan, B.: Almost square packing. In: Achterberg, T., Beck, J.C. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6697, pp. 196–209. Springer (2011), https://doi.org/10.1007/978-3-642-21311-3_19
32. Tang, T.Y., Beck, J.C.: CP and hybrid models for two-stage batching and scheduling. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 17th International Conference, CPAIOR 2020. Lecture Notes in Computer Science, vol. 12296, pp. 431–446. Springer (2020)
33. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**(3), 1109–1130 (2007)
34. Yang, D., Bambach, M., Cao, J., Duflou, J., Groche, P., Kuboki, T., Sterzing, A., Tekkaya, A., Lee, C.: Flexibility in metal forming. CIRP Annals **67**(2), 743–765 (2018)