RAIRO-Oper. Res. 42 (2008) 435–453
DOI: 10.1051/ro:2008027

RAIRO Operations Research
www.rairo-ro.org

A BRANCH-AND-PRICE-AND-CUT ALGORITHM FOR THE PATTERN MINIMIZATION PROBLEM

Cláudio Alves[1] and J.M. Valério de Carvalho[1]

Abstract. In cutting stock problems, after an optimal (minimal stock usage) cutting plan has been devised, one might want to further reduce the operational costs by minimizing the number of setups. A setup operation occurs each time a different cutting pattern begins to be produced. The related optimization problem is known as the Pattern Minimization Problem, and it is particularly hard to solve exactly. In this paper, we present different techniques to strengthen a formulation proposed in the literature. Dual feasible functions are used for the first time to derive valid inequalities from different constraints of the model, and from linear combinations of constraints. A new arc flow formulation is also proposed. This formulation is used to define the branching scheme of our branch-and-price-and-cut algorithm, and it allows the generation of even stronger cuts by combining the branching constraints with other constraints of the model. The computational experiments conducted on instances from the literature show that our algorithm finds optimal integer solutions faster than other approaches. A set of computational results on random instances is also reported.

Keywords. Pattern Minimization Problem, column generation, cutting planes, branch-and-bound, dual feasible functions.

Mathematics Subject Classification. 90C10, 90C57.

Introduction

Trim loss has been traditionally considered as the primary objective of cutting stock problems, but other costs may also be relevant. A setup cost, for example,

Received October 24, 2005; Accepted January 9, 2008.

[1] Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal;
claudio@dps.uminho.pt

is incurred each time we move from a pattern to a different one in a cutting plan. These changes take time. The knives must be repositioned to fit the next pattern. As a consequence, waste may be generated since various trial runs may be needed to reach the right positions. The problem of minimizing these setups is known as the Pattern Minimization Problem (PMP). In this paper, we propose an exact solution approach to solve it based on column generation, branch-and-bound and cutting planes.

The PMP is strongly NP-hard. Most of the literature reports on heuristic solution approaches. One of the oldest papers is due to Haessler [10,11]. It presents a sequential heuristic procedure. Other researchers tried to reduce the number of setups starting with a given solution by combining two or more patterns. Allwood and Goulimis [2] and Johnston [12,13] developed such algorithms. Foerster and Waescher [6] proposed recently the KOMBI heuristic, which allows many types of combinations. Chen *et al.* [4] proposed a slight variation of the basic simulated annealing algorithm to solve a non-linear model with two-sided demand constraints, different sizes of raw material, load balancing between the cutting machines and a combined trim loss and pattern minimization objective. The authors compared their algorithm to the standard simulated annealing using a small size instance (4 machine roll widths and 4 ordered items), and performed an extensive statistical analysis of the effects of the parameters. They obtained better objective values in less time. Teghem *et al.* [16] also used a simulated annealing algorithm for the problem of printing book covers with fixed and variable costs. Umetani *et al.* [17] proposed an iterated local search metaheuristic. Their algorithm was compared to the sequential heuristic procedure of Haessler and to the KOMBI heuristic. The computational experiments conducted indicate a comparable performance. Goulimis [9] solved the cutting stock problem with two-sided demand constraints using branch-and-bound and cutting planes. He focused on a set of small instances for which the complete set of cutting patterns can be generated in reasonable time. Setups were minimized afterwards, combining patterns using an approach inspired by Johnston [12].

As far as we know, only two exact approaches have been proposed in the literature. Vanderbeck [18] solved the PMP assuming a fixed number of stock rolls. He defined a model where columns are patterns with an associated multiplicity, and solved it using a branch-and-price-and-cut algorithm. The pricing subproblem is originally non-linear, but it can be linearized and solved as a sequence of integer bounded knapsack problems. Even if it improves the Linear Programming (LP) bound of the compact assignment formulation, the lower bound given by the model of Vanderbeck remains quite weak. In part, this is due to what the author called LP cheating, *i.e.*, columns with a large multiplicity getting preferentially fractional values in the LP solutions. For a set of 16 real-life instances, the author experienced an integrality gap of 33.5%, which was reduced to 13.8% by adding cutting planes to the LP relaxation. His branch-and-bound algorithm relies on a branching scheme based on 7 different rules. According to the author, these rules may fail in eliminating all the fractional solutions, and hence, the integrality gap may not be closed at the end of the algorithm.

Belov [3] considered the combined trim loss and pattern minimization problem. He used a model that was originally proposed by Vanderbeck [18], which is based on the Gilmore and Gomory formulation [8] for the standard Cutting Stock Problem (CSP). The model has general integer variables that represent the usage of the cutting patterns, and a binary variable for each cutting pattern with a value that depends on whether the pattern is used in the solution or not. Setups are modeled through these binary variables. For each valid cutting pattern, there are two columns in the model: one that represents the pattern, and another for the corresponding binary variable. Additionally, for each binary variable, there is also a defining constraint. Hence, in addition to a huge set of columns, the model has also a huge set of constraints. Belov showed that, using a bound on the pattern frequencies similar to the bound of Vanderbeck for the pattern multiplicities, the model becomes as strong as the model used by Vanderbeck in [18]. However, in order to make it suitable for practical computation, he removed the binary variables and the corresponding constraints, and used a non-linear objective function instead (with variable costs and fixed setup costs). The objective function is finally approximated by a linear one, and the resulting model is solved with branch-and-price. The model used by Belov has a very weak LP bound, and, as a consequence, his algorithm failed to solve 8 of the 16 instances used in [18]. Vanderbeck was unable to close the integrality gap for only 4 of them.

In this paper, we present an exact branch-and-price-and-cut algorithm for the PMP in the case where the number of stock rolls is bounded. In Section 1, some formulations are reviewed. We show how the largest pattern multiplicity can be reduced in the model of Vanderbeck, and we discuss the effects of this reduction. A new formulation based on arc flow variables is also introduced. In Section 2, we describe a new set of valid inequalities for the integer knapsack polytope derived using for the first time the dual feasible functions of Fekete and Schepers [5]. In Section 3, we clarify the details of our branch-and-bound procedure. Computational results are reported in Section 4, and followed by some final conclusions.

1. Integer linear programming formulations

In the PMP, we are given a set of m items of width w_i and demands of b_i units, $i = 1, \dots, m$, and a set of stock rolls with length W. Throughout the paper, we will assume all the data to be integer-valued. The problem consists in finding the cutting plan that minimizes the number of different patterns, while not using more than z_{CSP} stock rolls. Here, we consider that z_{CSP} is equal to the optimal solution of the corresponding CSP. In the sequel, we will assume that the items are sorted by decreasing widths, from w_1 the largest to w_m the smallest.

1.1. A compact formulation

The PMP can be formulated using a compact model with assignment variables and non-linear constraints as follows:

$$\min \sum_{k=1}^{z_{CSP}} y_k \tag{1}$$

$$\text{s.t.} \sum_{k=1}^{z_{CSP}} z_k x_{ik} = b_i,\ i = 1,\dots,m, \tag{2}$$

$$\sum_{k=1}^{z_{CSP}} z_k \le z_{CSP}, \tag{3}$$

$$z_k \le z_{CSP} y_k,\ k = 1,\dots,z_{CSP}, \tag{4}$$

$$\sum_{i=1}^{m} w_i x_{ik} \le W y_k,\ k = 1,\dots,z_{CSP}, \tag{5}$$

$$x_{ik} \ge 0 \text{ and integer},\ i = 1,\dots,m,\ k = 1,\dots,z_{CSP}, \tag{6}$$

$$y_k \in \{0,1\},\ k = 1,\dots,z_{CSP},\ z_k \ge 0 \text{ and integer},\ k = 1,\dots,z_{CSP}. \tag{7}$$

Variables x_{ik} denote the number of items i that are in pattern k, and z_k the number of times pattern k is used. Variables y_k are binary; they are equal to 1 only if pattern k is used. Since the objective is to minimize the sum of these y_k variables, and z_k are general integer variables, two patterns k_1 and k_2 with $k_1 \ne k_2$ will be necessarily different in an optimal solution. The demand constraints (2) involve non-linear terms. Constraint (3) limits the total number of rolls to z_{CSP}, the maximum number of stock rolls, while (5) represent the knapsack constraints. This compact model was first proposed by Vanderbeck in [18]. It has exactly $2z_{CSP} + mz_{CSP}$ variables.

1.2. Column generation reformulation

In [18], Vanderbeck proposed an alternative column generation model obtained by keeping in the master problem only constraints (2) and (3). This decomposition yields a non-linear pricing subproblem, which can be linearized by fixing one of its variables. Each column of the master is now a pattern replicated as many times as given by the column multiplicity. The master problem is stated as follows:

$$\min \sum_{p\in P} \sum_{n=1}^{ub(P_p)} \lambda_{pn} \tag{8}$$

$$\text{s.t.} \sum_{p\in P} \sum_{n=1}^{ub(P_p)} n a_{ip} \lambda_{pn} = b_i,\ i = 1,\dots,m, \tag{9}$$

$$\sum_{p\in P} \sum_{n=1}^{ub(P_p)} n \lambda_{pn} \le z_{CSP}, \tag{10}$$

$$\lambda_{pn} \in \{0,1\},\ p \in P,\ n = 1,\dots,ub(P_p). \tag{11}$$

The set of valid cutting patterns is denoted by P. Coefficients a_{ip} represent the number of items i in pattern p. The λ_{pn} variables denote the usage of the pattern p replicated n times, the so called pattern multiplicity. The maximum value of n for a pattern p, denoted as $ub(P_p)$, is equal to $\min\limits_{i=1,\ldots,m} \left\lfloor \frac{b_i}{a_{ip}} \right\rfloor$.

The pricing subproblem is the following non-linear knapsack problem:

$$\max\ n \left(\sum_{i=1}^{m} \pi_i x_i + \rho \right) \tag{12}$$

$$\text{s.t.} \ \sum_{i=1}^{m} w_i x_i \le W, \tag{13}$$

$$n x_i \le b_i, \ \ i = 1, \ldots, m, \tag{14}$$

$$n \in \{1, \ldots, n^{max}\}, \ \ x_i \ge 0 \text{ and integer}, \ \ i = 1, \ldots, m, \tag{15}$$

with a global upper bound on pattern multiplicities denoted by n^{max}. The vector of dual variables for (9) and (10) are denoted by π and ρ, respectively. Vanderbeck suggested the following value for n^{max}:

$$n^{max} = \min \left\{ z_{CSP} - \underline{z} + 1, \max_i \ b_i \right\}, \tag{16}$$

where $\underline{z}$ is a given lower bound on the minimum number of setups. The nonlinearities in (12)–(15) can be avoided if we consider instead a sequence of bounded integer knapsack problems with a fixed multiplicity n. In his paper, Vanderbeck noticed that, sometimes, it is not necessary to enumerate all the possible values of n, since an optimal solution to the linearized pricing subproblem may remain optimal for successive values of n. However, depending on the extra constraints that may be enforced in the master (branching constraints or cutting planes), a complete enumeration may be unavoidable.

1.3. Improving the column generation model of Vanderbeck

The model of Vanderbeck can be improved by using a bound on the total waste. Since the maximum number of stock rolls that can be used is at most equal to z_{CSP}, the waste in any solution must be at most equal to $l = z_{CSP}W - \sum_{i=1}^{m} w_i b_i$. Clearly, a column that corresponds to a replicated pattern with a total waste greater than l will never belong to any optimal integer solution. We can avoid generating these columns by enforcing the following constraint in the pricing subproblems:

$$W - \sum_{i=1}^{m} w_i x_i \le \left\lfloor \frac{l}{n} \right\rfloor. \tag{17}$$

As referred above, each column generation iteration consists in solving a sequence of knapsack problems for different values of n. For a given value of n, the knapsack problem with the additional constraint (17) consists in finding the best pattern that fits in the knapsack, and which is larger than or equal to $W - \left\lfloor \frac{l}{n} \right\rfloor$. The number of columns decreases compared to the original model of Vanderbeck. Expectably, the LP bound of the master problem should be stronger, but this is not the main value of this maximum waste constraint. This constraint helps in reducing the value of n^{max} and, hence, the number of knapsack problems that have to be solved to price the attractive columns.

Let L'_n and L_n denote the pricing subproblem (12)–(15) for a given value of n with and without constraint (17), respectively. Given a fixed n, L'_n and L_n are both integer linear programs. Whatever the value of n, L_n has always a feasible solution (at least the null solution is always valid), but this conclusion does not apply to L'_n. Indeed, it may be impossible to find a combination of items larger than or equal to $W - \left\lfloor \frac{l}{n} \right\rfloor$ that fits in the knapsack. Furthermore, since the knapsack problem is bounded (see constraint (14)), and given that the bounds depend on the value of n, if L'_n is infeasible for $n = n_1$, it will be infeasible for $n = n_1 + 1$, and so on. Hence, the formula for n^{max} can be rewritten as follows:

$$n^{max} = \min \left\{ z_{CSP} - \underline{z} + 1, \max_i \; b_i, n' \right\}, \tag{18}$$

with n' being the maximum value that n in L'_n can take in order for the problem to remain feasible. Generally, the value given by (18) is smaller than the multiplicity given by (16). In practice, it is not necessary to compute the exact value of n' a priori. Once the first infeasible pricing subproblem L'_n is found, the column generation iteration can be stopped. Thus, the value of n' can be updated dynamically. Similarly, when constraint (17) is considered, the maximum multiplicity $ub(P_p)$ of a pattern P_p becomes $\min \left\{ \min_{i=1,\dots,m} \left\lfloor \frac{b_i}{a_{ip}} \right\rfloor, \left\lfloor \frac{l}{W - \sum_{i=1}^{m} w_i a_{ip}} \right\rfloor \right\}$.

To evaluate the impact of the maximum waste constraint, we solved the LP relaxation of (8)–(11) for the instances in [18], with and without this constraint. The results are reported in Table 1. Column $Waste$ lists the waste generated in each instance by the optimal solution of the corresponding CSP. Column kp_{LP} reports the total number of knapsack problems solved. The number of generated columns ($cols_{LP}$), the maximum multiplicity among the added columns ($mult$) and the optimal LP solution (z_{LP}) are also reported. For these instances, the LP bound increases by 0.4% on average. On the other hand, the number of column generation subproblems solved is significantly reduced. While 880 knapsack problems are needed on average to reach the optimal LP solution, with the maximum waste constraint this value goes down to 585, a saving of almost 34%. Furthermore, there are 13% less columns generated and the maximum multiplicity of the newly generated columns decreases by almost 30%.

1.4. An arc flow formulation

The PMP can also be formulated as a flow problem over an acyclic digraph $G = (V, A)$, with three-indexes flow variables. A vertex in V corresponds to a

TABLE 1. Measuring the impact of the maximum waste constraint.

		Without max. waste const.				**With max. waste const.**			
Name	*Waste*	kp_{LP}	$cols_{LP}$	*mult*	z_{LP}	kp_{LP}	$cols_{LP}$	*mult*	z_{LP}
kT03	3088	110	30	31	4.77	99	26	31	4.77
kT05	102108	120	42	25	5.65	114	42	15	5.65
kT01	274	160	34	13	2.00	95	22	10	2.10
kT02	16575	468	114	13	15.93	461	112	13	15.93
kT04	36703	295	86	9	6.71	253	100	8	6.74
d16p6	36693	295	86	9	6.71	253	100	8	6.74
7p18	3826	264	44	84	3.74	155	32	56	3.74
d33p20	39905	869	160	13	6.05	583	136	8	6.18
12p19	526	643	98	13	2.88	489	90	12	2.89
d43p21	39764	1204	210	17	7.86	1018	202	13	7.86
kT06	1981	833	76	37	1.72	472	54	32	1.75
kT07	4990	1061	104	55	2.86	640	78	36	2.86
14p12	190500	1230	118	50	3.72	766	98	29	3.75
kT09	699	1577	130	94	3.65	1203	118	64	3.65
11p4	19000	1775	124	85	2.47	806	72	57	2.48
30p0	2562	3176	238	61	5.50	1946	194	36	5.51
avg.	31199.63	880.00	105.88	38.06	5.14	584.56	92.25	26.75	5.16

discrete position within the roll. Vertex 0 represents the leftmost border of the roll, and W the rightmost. Hence, we have $|V| = W + 1$. The set of arcs A is subdivided into n^{max} subsets A^n, one for each multiplicity. An arc (i, j) in A^n represents an item of width $j - i$ placed at a position i of the leftmost border of the roll in a pattern with multiplicity n. The unused portion of the rolls are represented by arcs (i, j) with $j - i = 1$. A pattern of multiplicity n is a path that starts at vertex 0 and ends at vertex W, using only arcs of A^n. The minimum number of different patterns is given by the minimum flow over G subject to some additional constraints. The model is stated as follows:

$$\min \sum_{n=1}^{n^{max}} z^n \tag{19}$$

$$\text{s.t.} -\sum_{(r,s)\in A^n} x^n_{rs} + \sum_{(s,t)\in A^n} x^n_{st} = \begin{cases} z^n, & \text{if } s = 0, \\ -z^n, & \text{if } s = W, \\ 0, & \text{otherwise,} \end{cases} \Bigg|\; n = 1, \dots, n^{max}, \tag{20}$$

$$\sum_{n=1}^{n^{max}} \sum_{(r,r+w_i)\in A^n} n x^n_{r,r+w_i} = b_i \ , \ i = 1, \dots, m, \tag{21}$$

$$x^n_{rs} \geq 0 \text{ and integer}, \ n = 1, \dots, n^{max}, \ \forall (r,s) \in A^n, \tag{22}$$

$$z^n \geq 0 \text{ and integer}, \ n = 1, \dots, n^{max}. \tag{23}$$

Variables x^n_{rs} denote the flow on arc (r,s) of A^n, while z^n, $n = 1,\dots,n^{max}$, represent the number of patterns with multiplicity n used. There is conservation of flow (constraints (20)) only among the arcs of the same set A^n. The maximum waste constraint discussed above is enforced by restricting in each A^n the set of arcs with unit size, those that model the unused space in the pattern. Note that although the variables of (8)–(11) are binary, the variables of the arc flow model (19)–(23) are general integer variables. Model (19)–(23) is an alternative formulation. Our branching scheme will be based on it.

Any pattern in (8)–(11) can be mapped into different sets of arc flows in G. In order to associate a single path in G to each specific column of the master (8)–(11), so that we can easily define sets of columns with a common property to branch on, we define the following mapping rule. A column with multiplicity n maps into an ordered sequence of arcs in A^n, starting at node 0, the left border of the roll. Items are converted into arcs in the order of decreasing widths. Hence, for two arcs in the same path, say (r,s) and (t,u) of A^n, if $s-r \ge u-t$, then $s \le t$. The arcs with unit size are left to the end of the roll.

2. New general cutting planes

In [18], Vanderbeck gave a set of valid superadditive inequalities for $S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \le b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \le b, \forall i\}$. Using the superadditive function $F^\gamma(z) = \max\left\{0, \lceil \frac{\gamma z}{b} \rceil - 1\right\}$, with $\gamma \in \{2,\dots,b\}$, he derived the inequalities:

$$\sum_i \left(\left\lceil \frac{\gamma a_i}{b} \right\rceil - 1 \right) x_i \le \gamma - 1, \tag{24}$$

and used them to strengthen the LP relaxation of (8)–(11). In this section, we present a new family of valid inequalities for S derived using dual feasible functions.

A function $f : [0,1] \to [0,1]$ is said to be dual feasible if $\sum_{x\in S} x \le 1 \Rightarrow \sum_{x\in S} f(x) \le 1$ holds for any finite set S of non-negative real numbers. Three dual feasible functions were used recently by Fekete and Schepers [5] to derive lower bounds for bin-packing problems. In the sequel, we will show that their functions are superadditive, and can consequently be used to generate valid inequalities for integer knapsack polytopes [15]. To the best of our knowledge, these functions were never used for this purpose.

In this paper, the dual feasible functions will be applied to the knapsack polytopes associated to different constraints of (8)–(11), namely the demand constraints (9), the constraint on the maximum number of rolls (10) and an additional total waste constraint. Note that these knapsack polytopes are different from the knapsack polytope associated to the pricing subproblems. The functions will be used to derive cutting planes that are also valid for the LP master problem.

All the dual feasible functions described in [5] are based on rounding. The first one ($u^{(k)}$, $k \in \mathbb{N}$) slightly improves a function proposed earlier by Lueker:

$$\begin{aligned} u^{(k)}: \quad & [0,1] \to [0,1] \\ & x \mapsto \begin{cases} x, & \text{for } (k+1)x \in \mathbb{Z}, \\ \frac{\lfloor (k+1)x \rfloor}{k}, & \text{otherwise.} \end{cases} \end{aligned}$$

Function $u^{(k)}$ is clearly nondecreasing. Its superadditivity is proved next.

Proposition 2.1. *For $k \in \mathbb{N}$, $u^{(k)}$ is superadditive over $[0,1]$.*

Proof: Let x, y, and $x+y$ be nonnegative real values in $\mathbb{R}_+ \cap [0,1]$. For $(k+1)(x+y) \notin \mathbb{Z}$, we have
$u^{(k)}(x+y) = \frac{\lfloor (k+1)(x+y) \rfloor}{k}$
$=\begin{cases} \frac{\lfloor (k+1)x \rfloor}{k} + \frac{\lfloor (k+1)y \rfloor}{k}, & \text{if } (k+1)(x+y) - (\lfloor (k+1)x \rfloor + \lfloor (k+1)y \rfloor) < 1, \\ \frac{\lfloor (k+1)x \rfloor}{k} + \frac{\lfloor (k+1)y \rfloor}{k} + \frac{1}{k}, & \text{if } (k+1)(x+y) - (\lfloor (k+1)x \rfloor + \lfloor (k+1)y \rfloor) \geq 1. \end{cases}$

Hence, the following cases may arise:

1. $(k+1)x \in \mathbb{Z}$ and $(k+1)y \notin \mathbb{Z}$: since $(k+1)(x+y) \notin \mathbb{Z}$ and $\frac{\lfloor (k+1)x \rfloor}{k} = \frac{(k+1)x}{k} > x$, we have $u^{(k)}(x) + u^{(k)}(y) = x + \frac{\lfloor (k+1)y \rfloor}{k} < u^{(k)}(x+y)$;
2. $(k+1)x \notin \mathbb{Z}$, $(k+1)y \notin \mathbb{Z}$ and $(k+1)(x+y) \notin \mathbb{Z}$: we have $u^{(k)}(x)+u^{(k)}(y) = \frac{\lfloor (k+1)x \rfloor}{k} + \frac{\lfloor (k+1)y \rfloor}{k} \leq u^{(k)}(x+y)$.

Additionally, we have to consider the following cases:

3. $(k+1)x \in \mathbb{Z}$ and $(k+1)y \in \mathbb{Z}$: this implies $(k+1)(x+y) \in \mathbb{Z}$, and $u^{(k)}(x) + u^{(k)}(y) = x + y = u^{(k)}(x+y)$;
4. $(k+1)x \notin \mathbb{Z}$, $(k+1)y \notin \mathbb{Z}$ and $(k+1)(x+y) \in \mathbb{Z}$: this implies $(k+1)(x+y) - (\lfloor (k+1)x \rfloor + \lfloor (k+1)y \rfloor) = 1$. Rearranging the terms, we get $x+y = \frac{\lfloor (k+1)x \rfloor}{k} + \frac{\lfloor (k+1)y \rfloor}{k} + \frac{1-(x+y)}{k}$. Since $x+y$ must be in $\mathbb{R}_+ \cap [0,1]$, we have finally $u^{(k)}(x) + u^{(k)}(y) = \frac{\lfloor (k+1)x \rfloor}{k} + \frac{\lfloor (k+1)y \rfloor}{k} \leq x + y = u^{(k)}(x+y)$. □

With the next proposition, we show that $u^{(k)}$ generates valid inequalities that are at least as strong as (24).

Proposition 2.2. *For $S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b,\ \forall i\}$ and $k \in \mathbb{N}$, the following inequality is equivalent to or dominates (24):*

$$\sum_i u^{(k)}\left(\frac{a_i}{b}\right) x_i \leq 1. \tag{25}$$

Proof: Let $z_i = \frac{a_i}{b}$ and, without loss of generality, assume that $k = \gamma - 1$. Inequalities (24) can be rewritten as follows $\sum_i \frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1} x_i \leq 1$. For $\gamma z_i \notin \mathbb{Z}$, we have $\frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1} = \frac{\lfloor \gamma z_i \rfloor}{\gamma - 1} = \frac{\lfloor (k+1) z_i \rfloor}{k} = u^{(k)}(z_i)$, being (25) equivalent to (24). On the other hand, for $\gamma z_i \in \mathbb{Z}$, we have $u^{(k)}(z_i) = z_i \geq \frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1}$, since $z_i \leq 1$, and $\frac{\lceil \gamma z_i \rceil - 1}{\gamma - 1} = \frac{\lceil (k+1) z_i \rceil - 1}{k} = \frac{(k+1) z_i - 1}{k} = z_i + \frac{z_i}{k} - \frac{1}{k}$, and so (25) dominates (24). □

The second dual feasible function $(u_1^{(\epsilon)})$ discussed in [5] formalizes a procedure of Martello and Toth [14] to derive the bin-packing lower bound L2. This function is also superadditive and nondecreasing, but the inequalities it generates for $S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b,\ \forall i\}$ can be weaker or stronger than (24), depending on the coefficients a_i and b of the original knapsack constraint. Let $\epsilon \in \left[0, \frac{1}{2}\right]$. The function is defined as follows:

$$u_1^{(\epsilon)} : \quad [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 0, & \text{for } x < \epsilon, \\ x, & \text{for } \epsilon \leq x \leq 1-\epsilon, \\ 1, & \text{for } x > 1-\epsilon. \end{cases}$$

Proposition 2.3. *Function $u_1^{(\epsilon)}$ is superadditive over $[0,1]$, for $\epsilon \in [0, \frac{1}{2}]$.*

Proof: Let $x,\ y,\ x+y \in \mathbb{R}_+ \cap [0,1]$ and consider the following three cases:

1. $x+y<\epsilon:$ in this case, we have $u_1^{(\epsilon)}(x+y) = 0$ and, since $x < \epsilon$ and $y < \epsilon$, the following holds $u_1^{(\epsilon)}(x) + u_1^{(\epsilon)}(y) = 0 = u_1^{(\epsilon)}(x+y)$;
2. $\epsilon \leq x+y \leq 1-\epsilon :$ then $x \leq 1-\epsilon$, $y \leq 1-\epsilon$, and consequently $u_1^{(\epsilon)}(x) + u_1^{(\epsilon)}(y) \leq x+y = u_1^{(\epsilon)}(x+y)$;
3. $x+y > 1-\epsilon :$ since $\epsilon \leq \frac{1}{2}$ and $x+y \in [0,1]$, if $x > 1-\epsilon$, then $y < \epsilon$ and $u_1^{(\epsilon)}(x) + u_1^{(\epsilon)}(y) = 1 = u_1^{(\epsilon)}(x+y)$, otherwise $u_1^{(\epsilon)}(x) + u_1^{(\epsilon)}(y) \leq x+y \leq 1 = u_1^{(\epsilon)}(x+y)$. □

Using two simple cases, the following example shows that inequalities (24) can dominate or be dominated by the following inequalities generated with $u_1^{(\epsilon)}$ for $S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b,\ \forall i\}$

$$\sum_i u_1^{(\epsilon)}\left(\frac{a_i}{b}\right) x_i \leq 1. \tag{26}$$

Example 2.1. *For $S = \{x \in \mathbb{N}^2 : 50x_1 + 52x_2 \leq 100\}$ and $\epsilon = 0.5$, (26) takes the form $0.5x_1 + x_2 \leq 1$. The value of $\frac{\lceil \frac{\gamma \times 52}{100} \rceil - 1}{\gamma - 1}$ (the coefficient of the left-hand side in (24)) is 1 only when $\gamma = 2$. In that case, $\frac{\lceil \frac{\gamma \times 50}{100} \rceil - 1}{\gamma - 1} = 0$. For $\gamma = 2$, (24) yields $x_2 \leq 1$, which is weaker than $0.5x_1 + x_2 \leq 1$. For all the other values of γ, we have $\frac{\lceil \frac{\gamma \times 52}{100} \rceil - 1}{\gamma - 1} < 1$. Furthermore, $\frac{\lceil \frac{\gamma \times 50}{100} \rceil - 1}{\gamma - 1} \leq 0.5$, $\forall \gamma \in [2, 100]$, and hence, (24) never leads to an inequality stronger than $0.5x_1 + x_2 \leq 1$.*

Consider now $S = \{x \in \mathbb{N}^2 : 10x_1 + 19x_2 \leq 100\}$. With $\gamma = 11$, (24) takes the form $0.1x_1 + 0.2x_2 \leq 1$. For $\epsilon \in [0, 0.1]$, we have $u_1^{(\epsilon)}(0.1) = 0.1$, while for $\epsilon \in]0.1, 0.5]$, we have $u_1^{(\epsilon)}(0.1) = 0$. Similarly, for $\epsilon \in [0, 0.19]$, we have $u_1^{(\epsilon)}(0.19) = 0.19$, while for $\epsilon \in]0.19, 0.5]$, we have $u_1^{(\epsilon)}(0.19) = 0$. The following inequalities hold for all $\epsilon \in [0, 0.5]$: $u_1^{(\epsilon)}(0.1) \leq 0.1$ and $u_1^{(\epsilon)}(0.19) \leq 0.19$. As a *consequence, the dual feasible function $u_1^{(\epsilon)}$ cannot generate any cut stronger than* $0.1x_1 + 0.19x_2 \leq 1$, *i.e., the original inequality.* □

The third dual feasible function in [5] is denoted by $u_2^{(\epsilon)}$, with ϵ in $[0, \frac{1}{2}($:

$$u_2^{(\epsilon)} : \quad [0,1] \to [0,1]$$
$$x \mapsto \begin{cases} 0, & \text{for } x < \epsilon, \\ \frac{1}{\lfloor \epsilon^{-1} \rfloor}, & \text{for } \epsilon \leq x < \frac{1}{2}, \\ \frac{1}{2}, & \text{for } x = \frac{1}{2}, \\ 1 - \frac{\lfloor (1-x)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor}, & \text{for } x > \frac{1}{2}. \end{cases}$$

The following example shows that $u_2^{(\epsilon)}$ is not superadditive for all the values of ϵ in $[0, \frac{1}{2}($. However, $u_2^{(\epsilon)}$ is superadditive for a smaller interval for ϵ, as we show in the next proposition.

Example 2.2. *Let ϵ in $u_2^{(\epsilon)}$ be equal to 0.1. For $x = 0.15$ and $y = 0.15$, we have $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = 0.2 > u_2^{(\epsilon)}(x+y) = 0.1$, which clearly violates the superadditivity condition.* □

Proposition 2.4. *For $\epsilon \in) \frac{1}{4}, \frac{1}{2}($, $u_2^{(\epsilon)}$ is superadditive over $[0,1]$.*

Proof: Consider the following four cases:

1. $x + y < \epsilon$: then $x < \epsilon$, $y < \epsilon$ and $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = 0 = u_2^{(\epsilon)}(x+y)$;
2. $\epsilon \leq x + y < \frac{1}{2}$: then $x < \frac{1}{2}$ and $y < \frac{1}{4}$ (or vice versa). We have $u_2^{(\epsilon)}(x) \leq \frac{1}{\lfloor \epsilon^{-1} \rfloor}$, $u_2^{(\epsilon)}(y) = 0$, since $\epsilon > \frac{1}{4}$, and hence $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) \leq \frac{1}{\lfloor \epsilon^{-1} \rfloor} = u_2^{(\epsilon)}(x+y)$;
3. $x + y = \frac{1}{2}$: if $x = \frac{1}{2}$, then $y = 0$ (or vice versa), and $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = \frac{1}{2} = u_2^{(\epsilon)}(x+y)$. If $x < \frac{1}{2}$, then $y \leq \frac{1}{4}$, (or vice versa); $u_2^{(\epsilon)}(x) \leq \frac{1}{\lfloor \epsilon^{-1} \rfloor} \leq \frac{1}{2}$, $u_2^{(\epsilon)}(y) = 0$, and hence $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) \leq \frac{1}{2} = u_2^{(\epsilon)}(x+y)$;
4. $x + y > \frac{1}{2}$: with $\epsilon \in) \frac{1}{4}, \frac{1}{2}($, we have $\frac{2}{\lfloor \epsilon^{-1} \rfloor} \leq 1 - \frac{\lfloor (1-(x+y))\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} = u_2^{(\epsilon)}(x+y) \leq 1$, and hence, superadditivity is proven for the cases where x and y are both strictly less than $\frac{1}{2}$. If x and y are both equal to $\frac{1}{2}$, we have $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = 1 = u_2^{(\epsilon)}(x+y)$. For $x > \frac{1}{2}$ and $y < \epsilon$, we have $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = 1 - \frac{\lfloor (1-x)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} \leq 1 - \frac{\lfloor (1-(x+y))\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} = u_2^{(\epsilon)}(x+y)$. The remaining case is when $x > \frac{1}{2}$ and $\epsilon \leq y < \frac{1}{2}$. We have $\lfloor (1-(x+y))\epsilon^{-1} \rfloor = \lfloor (1-x)\epsilon^{-1} - y\epsilon^{-1} \rfloor \leq \lfloor (1-x)\epsilon^{-1} - 1 \rfloor = \lfloor (1-x)\epsilon^{-1} \rfloor - 1$, and hence $u_2^{(\epsilon)}(x) + u_2^{(\epsilon)}(y) = 1 - \frac{\lfloor (1-x)\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor} + \frac{1}{\lfloor \epsilon^{-1} \rfloor} \leq 1 - \frac{\lfloor (1-(x+y))\epsilon^{-1} \rfloor}{\lfloor \epsilon^{-1} \rfloor}$. □

Once more, $u_2^{(\epsilon)}$ is clearly nondecreasing and, for $\epsilon \in) \frac{1}{4}, \frac{1}{2}($, the following inequalities it generates for $S = \{x \in \mathbb{N}^n : \sum_i a_i x_i \leq b,\ a \in \mathbb{N}^n,\ b \in \mathbb{N},\ a_i \leq b,\ \forall i\}$ can dominate or be dominated by (24), as it is shown in Example 2.3:

$$\sum_i u_2^{(\epsilon)} \left(\frac{a_i}{b} \right) x_i \leq 1. \tag{27}$$

Example 2.3. *For* $S = \{x \in \mathbb{N}^2 : 10x_1 + 19x_2 \leq 100\}$, $u_2^{(\epsilon)}$ *with* $\epsilon \in]\frac{1}{4}, \frac{1}{2}[$ *produces no useful valid inequality, since* $u_2^{(\epsilon)}(0.1) = 0$ *and* $u_2^{(\epsilon)}(0.19) = 0$, $\forall \epsilon \in]\frac{1}{4}, \frac{1}{2}[$.

For $S = \{x \in \mathbb{N}^2 : 270x_1 + 745x_2 \leq 1000\}$ *and* $\epsilon = 0.26$, $u_2^{(\epsilon)}$ *yields* $\frac{1}{3}x_1 + x_2 \leq 1$, *which is stronger than any inequality (24). The proof is similar to the one given in Example 2.1. The value of* $\frac{\lceil \frac{\gamma \times 745}{1000} \rceil - 1}{\gamma - 1}$ *is 1 only when* $\gamma \in \{2, 3\}$. *In those cases,* $\frac{\lceil \frac{\gamma \times 270}{1000} \rceil - 1}{\gamma - 1} = 0$, *and hence for* $\gamma \in \{2, 3\}$, *(24) yields a cut that is weaker than* $\frac{1}{3}x_1 + x_2 \leq 1$. *For all the other values of* γ, *we have* $\frac{\lceil \frac{\gamma \times 745}{1000} \rceil - 1}{\gamma - 1} < 1$. *Furthermore,* $\frac{\lceil \frac{\gamma \times 270}{1000} \rceil - 1}{\gamma - 1} \leq \frac{1}{3}$, $\forall \gamma \in [2, 1000]$, *and hence, (24) never leads to an inequality stronger than* $\frac{1}{3}x_1 + x_2 \leq 1$. □

3. The branch-and-price-and-cut algorithm

In this section, we describe the main features of our branch-and-price-and-cut algorithm. The whole procedure can be summarized as follows. At each node of the branching tree, we solve the LP relaxation of (8)–(11) with the maximum waste constraint described above. A cutting plane procedure is then invoked to generate the cuts presented in the previous section (and others described below) to improve the LP bound. The resulting LP optimal solution is then converted in an arc flow representation, branching constraints are defined on these flow variables and converted back into constraints in (8)–(11). No sophisticated rounding scheme is used along the solution process. A depth-first strategy is used to privilege a faster improvement of the incumbent.

The first restricted LP master related to (8)–(11) is initialized with an artificial column with high cost, and the patterns that are in the optimal basis of the corresponding CSP. The master problem is solved using column generation. To correctly account for the dual values of the branching constraints in the pricing subproblems, we have to know the exact position of the items in the knapsack. For this purpose, we used a dynamic programming algorithm to solve the pricing subproblems. Before describing the cutting plane procedure, we introduce first the details of our branching scheme, because cutting planes are not only derived from the constraints of (8)–(11), but also from branching constraints. Whenever cutting planes are not enough to close the integrality gap, we resort to branch-and-bound.

A desirable requirement for a branching scheme is to be compatible with the subproblem. A branching scheme should also guarantee that all the fractional solutions are eliminated in order for the algorithm to end up with an optimal integer solution. In [18], Vanderbeck proposed a list of branching rules based on hyperplanes that did not satisfy this latter condition. Hence, with his algorithm, the integrality gap may not be closed. Here, we consider branching on the original arc flow variables of (19)–(23). Our branching scheme does not induce any intractable modification to the pricing subproblem, and it guarantees that the algorithm finds an optimal integer solution after a finite number of steps.

When the solution of the LP master is fractional and the corresponding node cannot be pruned by bounding, we convert the fractional solution into a set of arc flows using the mapping rules described in Section 1.4. If the mapping yields no fractional arc flow variables, then, by the flow decomposition property, we can recover an integer solution to (8)–(11) with a cost equal to the LP bound. In those cases, obviously, we do not branch. Branching is necessary only if there is at least one fractional arc flow variable. The variable x^n_{rs} with a fractional value and largest multiplicity (with a larger n) corresponding to the leftmost arc (with a smaller r) is the one selected for branching (ties are broken by choosing the arc associated to the largest item). Two nodes are created with the following branching constraints: $x^n_{rs} \leq \lfloor x^n_{rs} \rfloor$ and $x^n_{rs} \geq \lceil x^n_{rs} \rceil$. These constraints are easily enforced back in the LP master. All the columns with multiplicity n that map into a path with an arc (r, s) have $a + 1$ coefficient in the branching constraint. When certain of these branching constraints are imposed, the LP lower bound can eventually be tightened. Consider, for example, the branching constraint $x^n_{rs} \geq lb$ with $W - s < w_i$, for all $i = 1, \ldots, m$, *i.e.*, no item can be placed at position s. That means that part of the total waste (precisely $lb \times n(W - s)$) will surely be concentrated on patterns with multiplicity n. Hence, among the patterns with multiplicities $n' \neq n$, only those with a waste equal to or lower than $\left\lfloor \frac{l - lb \times n(W-s)}{n'} \right\rfloor$ need to be generated in this node and its descendants. The patterns that are in the master and violate this condition can be just removed.

There is no symmetry in the branch-and-bound tree. By branching on the flow variables of the original arc flow model, the resolution of (8)–(11) with branch-and-price remains possible in practice since the complexity of the pricing subproblems is unchanged. As happens with the Gilmore and Gomory model for the standard CSP [8], model (8)–(11) has no symmetry. A setup associated to a specific pattern of multiplicity n has a single representation, and hence, there is only one possible representation for each complete solution to (8)–(11). Furthermore, a branching constraint on a variable x^n_{rs} converts into a unique constraint on the λ_{pn} variables of the LP master problem. Consequently, the solution space of two disjoint nodes is mutually exclusive.

Cutting planes of type (25), (26) and (27) are derived from different constraints: the constraint on the maximum number of rolls (10), the demand constraints (9) and the waste constraint. In fact, instead of using the exact constraint on the number of rolls as defined in (10), we generate cuts based on the following linear combination of constraints: $\sum_{p \in P} \sum_{n \in N_p} (n - 1)\lambda_{pn} \leq z_{CSP} - LB^q$, where N_p stands for the set of multiplicities for pattern p and LB^q is the lower bound on the number of setups at a node q of the branch-and-bound tree (note that $n - 1 \geq 0$). This constraint results from the combination of (10) with $\sum_{p \in P} \sum_{n \in N_p} \lambda_{pn} \geq LB^q$. Note that LB^q is always the best known lower bound for node q. As a consequence, some of the cuts may be valid for that node q and its descendants, but not for the other nodes, those with a smaller lower bound on the number of setups. Applying dual feasible functions to linear combinations of constraints may lead to stronger cuts. In the latter case, the coefficients of the left hand side of (10) are decreased by only one unit, while its right hand side is decreased by a greater value. The ratio between the coefficients of the constraint and its right hand side increases. The following example shows that combining constraints may yield stronger cuts.

Example 3.1. *Consider the following two inequalities* $A1 : 7x_1 + 8x_2 + 5x_3 \leq 10$ *and* $A2 : x_1 + x_2 + x_3 \geq 2$*, and assume that all the variables are binary variables. If both these inequalities are valid for a given integer linear program, then the inequality* $A3 : 6x_1 + 7x_2 + 4x_3 \leq 8$ *is also valid. This inequality is obtained by subtracting the coefficients of* $A2$ *from the coefficients of* $A1$*. The inequalities* $A1$ *and* $A3$ *can be re-written respectively as follows:* $\frac{7}{10}x_1 + \frac{4}{5}x_2 + \frac{1}{2}x_3 \leq 1$ *and* $\frac{3}{4}x_1 + \frac{7}{8}x_2 + \frac{1}{2}x_3 \leq 1$*. Applying the dual feasible function* $u^{(k)}$ *with* $k = 3$ *to* $A1$*, we would get* $\frac{2}{3}x_1 + x_2 + \frac{1}{2}x_3 \leq 1$*. Applying the same dual feasible function to* $A3$ *yields the following stronger inequality:* $\frac{3}{4}x_1 + x_2 + \frac{1}{2}x_3 \leq 1$. □

Using the same principle, we combine the "greater than or equal to" branching constraints at a node q with the demand constraints (9) to derive the following constraints: $\sum_{p \in P} \sum_{n \in N_p} (na_{ip} - \sum_{(r,r+w_i,n) \in H^q} g_{pn}^{r,r+w_i})\lambda_{pn} \leq b_i - \sum_{(r,r+w_i,n) \in H^q} lb_{r,r+w_i}^n$, $i = 1, \dots, m$. The set of "greater than or equal to" branching constraints at a node q is denoted by H^q, while g_{pn}^{rs} are binary coefficients equal to 1 if column pn maps into a path with the arc (r, s) of A^n, and equal to 0 otherwise. The right hand side of the "greater than or equal to" branching constraints on an arc (r, s) with multiplicity n in H^q is equal to lb_{rs}^n. The deeper a node is in the tree, the stronger will be these cuts.

An additional set of valid inequalities is derived from the following total waste constraint: $\sum_{p \in P} \sum_{n \in N_p} l_{pn}\lambda_{pn} \leq l$, where l_{pn} is the total amount of waste for n copies of pattern p. This constraint is implicit in the master. It is induced by the demand constraints and the constraint on the number of rolls. However, it helps in deriving violated inequalities.

Violated inequalities are found by enumerating on the value of the parameters of the dual feasible functions. For $u^{(k)}$, we generate valid inequalities for successive values of k. For $u_1^{(\epsilon)}$ and $u_2^{(\epsilon)}$, the values of ϵ depend on the list of item widths. The dual feasible functions are applied to the normalized coefficients of the constraints referred to above. If the current solution violates the generated inequality, we compare the deviation with that of the most violated cutting plane found so far. Only the most violated cut is added to the LP master in each iteration. The cutting plane procedure repeats until there is no more inequalities violated in more than 0.0001. Further details concerning the separation procedures can be found in [1].

To accelerate branch-and-bound, the nodes are inspected to anticipate infeasibility or to prune them by computing lower bounds. Consider a node q of the branch-and-bound tree with a set H^q of "greater than or equal to" branching constraints, and let $H_1^q = \{(r,s,n) \in H^q : W - s < w_m\}$, with w_m being the width of the smallest item. If the following holds $\sum_{(r,s,n)\in H_1^q} n \times lb_{rs}^n \times (W-s) > l$, node q can be pruned, since the branching constraints force a solution with more waste than the maximum waste of the optimal CSP solution. Based on the branching constraints at a node q, we can calculate a lower bound on the number of different patterns in two different ways. Let H_{2n}^q be the set of "greater than or equal to" branching constraints at node q, such that two triplets (r_1,s_1,n) and (r_2,s_2,n) belong to H_{2n}^q if they both belong to H^q and $s_1-r_1+s_2-r_2 > W$. The set H_{2n}^q is composed by the triplets $(r,s,n) \in H^q$ with $s-r > \frac{W}{2}$ (let w' be the width of the smallest of these arcs), and a triplet (r_1,s_1,n) of H^q with the greatest associated lb_{rs}^n at node q and such that $\frac{W}{2} \geq s_1 - r_1 > W - w'$. Clearly, the arcs in H_{2n}^q will appear in different patterns. If $\sum_{n=1}^{n^{max}} \sum_{(r,s,n)\in H_{2n}^q} lb_{rs}^n \geq z_{inc}$, with z_{inc} being the value of the current incumbent, node q can be pruned, since it surely leads to a non-improving solution. On the other hand, if $\sum_{n=1}^{n^{max}} \sum_{(r,s,n)\in H_{2n}^q} n \times lb_{rs}^n > z_{CSP}$, node q will be infeasible, and it can therefore be pruned in anticipation. A second lower bound can be computed as follows: $\sum_{n=1}^{n^{max}} \max\limits_{(r,s,n)\in H^q} lb_{rs}^n$, since $\max\limits_{(r,s,n)\in H^q} lb_{rs}^n$ is a lower bound on the number of patterns with multiplicity n. The node can be pruned by comparing this bound with z_{inc}, and by comparing $\sum_{n=1}^{n^{max}} \max\limits_{(r,s,n)\in H^q} n \times lb_{rs}^n$ with z_{CSP}.

4. Computational results

To evaluate the performance of our algorithm, we conducted two sets of computational experiments with a 3.0 GHz Pentium IV computer with 512 MB of RAM and CPLEX 6.5 with default settings. The first set of instances was used by Vanderbeck in [18]. Some of them come from real-life problems. The computational results show that our approach improves in some aspects the exact method proposed by Vanderbeck. Additionally, we performed some experiments on randomly generated instances.

Table 2 summarizes the computational results obtained with the instances used in [18]. The columns have the following meaning: $Name$ identifies the instance; m is the number of different items; sp_{LP} is the number of subproblems solved at the root node; $cols_{LP}$ is the number of columns generated at the root node; sp_{BB} is the total number of subproblems solved at the nodes of the branch-and-bound tree, excluding the root node; $cols_{BB}$ is the number of columns generated during branch-and-bound; nod_{BB} is the number of nodes of the branch-and-bound tree (excluding the root node); $cuts$ is the total number of cutting planes added; BBP is the initial lower bound obtained by solving the corresponding bin-packing instance; z_{LP}^{bc} is the LP optimum before any cut is added; z_{LP}^{ac} is the LP optimum after cuts are applied; LB is the best lower bound obtained in the course of the algorithm; UB is the value of the best incumbent; CSP is the number of different patterns in the solution of the corresponding standard CSP; K is the number of stocks rolls that minimizes trim loss; t_{LP} is the total computing time spent at the root node; t_{BB} is the total time spent at the nodes of the branch-and-bound tree, after the root node; t_{TOT} is the total computing time. The maximum computing time was limited to 2 h.

The algorithm solved successfully 13 of the 16 instances, while Vanderbeck only closed the optimality gap for 12 of them. For these 12 instances, on average, our algorithm needs much less branching nodes. If we exclude the root node, Vanderbeck needed 98 nodes to solve these instances, while we only need 55.3 nodes. Remember that we do not use any rounding heuristic with which a better incumbent may probably be found faster. The LP bounds obtained after adding our cutting planes are better than the ones obtained with the cutting planes described by Vanderbeck. For kT03, for example, we are able to close the integrality at the root node. The lower bound of 5 units obtained by Vanderbeck is improved by 10%. On average, at the root node, the values of the LP optima are improved by 21.5%. For 11p4, this improvement reaches 26.1%.

Our algorithm was further tested on a broad range of randomly generated instances. To generate them, we used CUTGEN1 [7]. A total of 3600 instances divided in 36 groups of 100 instances were generated for different problem sizes (m = 20, 30 and 40), different average demands (d = 10, 20 and 30), a single stock length (W = 1000), and various relative widths of the items compared to the stock lengths. The width of the smallest item (v_1) varies between 1 and 30% of the stock length ($v_1 \in \{1, 10, 20, 30\}$), while the width of the largest item (v_2) is always at most 80% of the stock length. We used a seed equal to 1994, and stopped the execution after 10 min of branch-and-bound. A complete list of the results is available in [1].

All the instances with $m = 20$ and an average demand of 10 units per item width are solved to optimality. For the other instances with $m = 20$, we are not able to close the integrality gap for only 11% of the instances. The corresponding average gap is not greater than 3%. The most difficult instances are those with $d = 30$. In fact, the larger the demands, the larger will be the multiplicities, and the larger will be the number of subproblems that will have to be solved. Computing times increase naturally with the average demand, and other parameters like the relative width of the smallest item, for example. Applying the cutting planes yields an improvement of 17.5% of the LP bound. The average computing time is almost 2.5 min. The percentage of instances with $m = 30$ solved to optimality is 57.6%. For these instances, the cutting planes improve the LP bound by 19.8%. The optimality gap increases to 8.9%. The average computing time is slightly greater than 6 min. Only 33.5% of the instances with $m = 40$ are optimally solved, which amounts to nearly 400 instances. For these instances, the average optimality gap is 12.1% and the average computing time is almost 9 min.

TABLE 2. Computational results for instances from the literature.

$Name$	m	sp_{LP}	$cols_{LP}$	sp_{BB}	$cols_{BB}$	nod_{BB}	$cuts$	LB_0	z_{LP}^{bc}	z_{LP}^{ac}	LB	UB	CSP	z_{CSP}	t_{LP}	t_{BB}	t_{TOT}
kT03	7	70	64	0	0	0	46	3	4.77	5.50	6	6	6	66	0.1	0.0	0.1
kT05	10	55	46	157	87	65	62	4	5.65	8.00	9	9	11	47	2.8	8.4	11.2
kT01	5	36	35	14	14	1	57	1	2.1	2.61	3	3	6	14	0.3	0.1	0.4
kT02	24	121	119	93	61	26	69	13	15.93	18.00	18	18	20	66	0.6	1.2	1.8
kT04	16	201	166	356	281	88	113	6	6.74	7.88	9	9	16	38	2.5	11.2	13.7
d16p6	16	202	173	199	144	53	107	6	6.74	7.90	9	9	14	38	6.0	6.2	12.2
7p18	7	65	53	587	437	193	170	2	3.74	4.90	6	6	8	91	2.6	43.3	45.9
d33p20	23	147	146	1253	1202	114	123	5	6.18	6.70	8	8	17	29	17.1	209.0	226.1
12p19	12	98	96	1035	1004	63	167	2	2.89	3.90	5	5	15	23	15.8	135.6	151.4
d43p21	32	211	207	767	732	69	177	7	7.86	8.72	10	10	26	40	31.3	194.5	225.9
kT06	9	160	131	645	602	11	309	1	1.75	2.78	4	4	15	51	144.3	758.8	903.0
kT07	11	157	155	3458	3367	168	477	2	2.86	3.64	5	5	16	65	69.2	3393.5	3462.7
14p12	14	140	119	112	110	5	178	2	3.75	4.22	5	5	18	56	40.7	34.3	75.0
kT09	14	138	136	3656	3574	175	313	2	3.65	4.95	5	6	24	110	129.0	7071.3	7200.3
11p4	11	225	223	2135	2068	65	372	1	2.48	3.91	4	5	24	101	197.4	7002.9	7202.3
30p0	26	313	311	2051	2043	28	294	4	5.51	6.66	7	8	28	90	265.9	6935.6	7201.5
avg.	14.8	146.2	136.3	1032.4	982.9	70.3	189.6	3.8	5.16	6.27	7.1	7.3	16.5	57.8	57.9	1613.0	1670.8

5. Conclusion

In this paper, we described a branch-and-price-and-cut algorithm for the Pattern Minimization Problem. A new arc flow formulation was proposed, and new valid inequalities for the integer knapsack polytope were presented. These inequalities were used to strengthen the continuous bound of the column generation model. We used dual feasible functions for the first time to derive these cutting planes. We gave various formal proofs showing that some of these functions discussed in the literature are superadditive, and hence, that valid inequalities can be obtained with them. Stronger cuts were computed by using linear combinations of constraints instead of the original ones. Various computational experiments were performed on instances from the literature, and randomly generated instances. Our algorithm improved the results obtained by other exact approaches.

Acknowledgements. We thank the anonymous referees for their constructive comments, which led to a clearer presentation of the material.
This work was partially supported by the Portuguese Science and Technology Foundation (POS_C/57203/EIA/2004) and by the Algoritmi Research Center of the University of Minho (Industrial and Systems Engineering Group).

References

[1] C. Alves, *Cutting and packing: problems, models and exact algorithms.* Ph.D. Thesis, Universidade do Minho (2005).
[2] J.M. Allwood and C.N. Goulimis. *Reducing the number of patterns in one-dimensional cutting stock problems.* Technical report, Electrical Engineering Department, Imperial College, London (1988).
[3] G. Belov. *Problems, models and algorithms in one- and two- dimensional cutting.* Ph.D. Thesis, Dresden University (2003).
[4] C.-L.S. Chen, S.M. Hart, and W.M. Tham. A simulated annealing heuristic for the one-dimensional cutting stock problem. *Eur. J. Oper. Res.* **93** (1996) 522–535.
[5] S. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Math. Program.* **91** (2001) 11–31.
[6] H. Foerster and G. Waescher. Pattern reduction in one-dimensional cutting stock problems. *Int. J. Prod. Res.* **38** (2000) 1657–1676.
[7] T. Gau and G.Waescher. CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem. *Eur. J. Oper. Res.* **84** (1995) 572–579.
[8] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem. *Oper. Res.* **9** (1961) 849–859.
[9] C. Goulimis, Optimal solutions for the cutting stock problem. *Eur. J. Oper. Res.* **44** (1990) 197–208.
[10] R.W. Haessler, A heuristic programming solution to a nonlinear cutting stock problem. *Manage. Sci.* **17** (1971) 793–802.
[11] R.W. Haessler. Controlling cutting pattern changes in one-dimensional trim problems. *Oper. Res.* **23** (1975) 483–493.
[12] R.E. Johnston, Rounding algorithms for cutting stock problems. *Asia-Pac. Oper. Res. J.* **3** (1986) 166–171.

[13] R.E. Johnston. Cutting patterns and cutter schedules. *Asia-Pac. Oper. Res. J.* **4** (1987) 3–14.

[14] S. Martello and P. Toth, *Knapsack Problems*. Wiley, New York (1990).

[15] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*. Wiley, New York (1988).

[16] J. Teghem, M. Pirlot, and C. Antoniadis, Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. *J. Comput. Appl. Math.* **64** (1995) 91–102.

[17] S. Umetani, M. Yagiura, and T. Ibaraki, One-dimensional cutting stock problem to minimize the number of different patterns. *Eur. J. Oper. Res.* **146** (2003) 388–402.

[18] F. Vanderbeck. Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Oper. Res.* **48** (2000) 915–926.