New stabilization procedures for the cutting stock problem

François Clautiaux*, Cláudio Alves†, and José Valério de Carvalho†
* Université des Sciences et Technologies de Lille, LIFL UMR CNRS 8022, INRIA
Bâtiment INRIA, Parc de la Haute Borne, 59655 Villeneuve d'Ascq, France
francois.clautiaux@univ-lille1.fr
† Departamento de Produção e Sistemas, Escola de Engenharia,
Universidade do Minho, 4710-057 Braga, Portugal
{claudio,vc}@dps.uminho.pt

May 9, 2008

Abstract

In this paper, we deal with a column generation based algorithm for the classical cutting stock problem. This algorithm is known to have convergence issues, which are addressed in this paper. Our methods are based on the fact that there are interesting characterizations of the structure of the dual problem, and that a large number of dual solutions are known. First we describe methods based on the concept of *dual cuts*, proposed by Valério de Carvalho (2005). We introduce a general framework for deriving cuts, and we describe a new type of dual cuts, which exclude solutions that are linear combinations of some other known solutions. We also explore new lower and upper bounds for the dual variables, and we present a procedure to set some of these variables to zero. Then we show how the prior knowledge of a good dual solution helps improving the results. It tightens the bounds around the dual values, and makes the search converge faster if a solution is sought in its neighborhood first. A set of computational experiments on very hard instances are reported at the end of the paper. They confirm the effectiveness of the methods proposed.

1 Introduction

The cutting stock problem (CSP) consists in finding the minimum number of identical stock rolls needed to cut a set of items of different sizes. It belongs to the class of *Cutting and Packing* problems as a *single stock-size cutting stock problem* [32]. Each item has a given demand, which corresponds to the number of times it has to be cut from the stock rolls.

Many researchers have used Dantzig-Wolfe decomposition [10] and *column generation methods* for solving the CSP (see [15, 16] for the first attempt). The integer program is decomposed into a restricted master problem initialized with a set of columns, and optimized to determine the value of the dual variables. The dual information is passed to a subproblem that evaluates if there is still any column that can be added to the master problem and improve the current solution. If there is such a column, the master problem is reoptimized, otherwise the process stops.

Column generation processes are known to have convergence issues. The dual variables may oscillate from one iteration to the next [12], and primal degeneracy may also arise [23]. Convergence of column generation has been the topic of many contributions. One of the first approaches was the *Boxstep* method of Marsten *et al.* [24]. Their method tries to stabilize and accelerate the column generation process by guiding the dual variables, imposing box constraints on the solutions of the restricted master. Kallehauge *et al.* [21] used a similar scheme to solve the vehicle routing problem with time windows, but instead of considering fixed-size boxes as in [24], they allowed the boxes to be dynamically updated. In both [24] and [21], the dual variables cannot lie outside the boundaries defined by the box constraints. In [12], du Merle *et al.* relaxed this constraint. The dual variables may take values outside the boxes, but there is a proportional penalty if that situation occurs. In the primal, this approach consists in adding weighted

and bounded slack and surplus variables to the original constraints of the problem. The authors report good results for transportation and location problems.

Other approaches that keep the solution process within the framework of linear programming have been described in the literature. Kim *et al.* [22], for example, considered a linear penalty method that can be extended to column generation ([26]). As an alternative, many nonlinear methods can be used to stabilize column generation such as the bundle methods [18] and the analytic centers cutting plane methods [17], for example.

Degraeve and Peeters [11] solved the CSP with column generation using an hybrid procedure that relies on a subgradient method. Initially, the dual values of the restricted master are updated using subgradient optimization, and columns are generated based on these values. Simplex is used at a second stage to optimize the master problem with the new columns. The subgradient method allows for a fast update of the dual values, and in practice, their approach accelerates the column generation process.

One of the most promising approaches to stabilized column generation has been proposed by Valério de Carvalho in [31]. The method consists in adding a polynomial number of *dual cuts* (primal columns) to the restricted master problem. In [31], the author proposed a set of *weak* dual-optimal inequalities for the CSP, which do not cut any optimal solution of the dual problem ([4]). Since then, the method has been used successfully to accelerate other cutting problems [1, 29]. In [2], Alves and Valério de Carvalho showed how to solve the multiple length CSP with branch-and-price using these dual cuts in all the nodes of the branch-and-bound tree. In this paper, we explore the dual cuts of Valério de Carvalho from a different point of view. We propose new valid dual cuts for the CSP based on new theoretical results, and we devise new procedures to stabilize column generation processes.

The cuts of [31] can be seen as a restriction of the dual space to a subset of non-dominated solutions sharing a special structure. The method is based on the fact that there is always an optimal dual solution whose coefficients can be obtained by applying a non-decreasing and *superadditive* function to the sizes of the items. In this paper, we introduce new dual cuts, which take into account the fact that a maximal solution has to be *symmetric* in addition to increasing and superadditive.

As they are defined in [31], the dual cuts only exclude solutions that are dominated by another single solution. Another type of cuts may be applied. It is a classical result of linear programming that any solution that is the midpoint of two other solutions cannot lead to improved results, even if it is not dominated by any of these two solutions separately. Consequently, finding properties of non-dominated solutions may lead to new effective dual cuts, if they can be expressed by the mean of a small set of linear constraints. Our approach is the following. First we show general results, which allow one to derive cuts from a valid dual solution. Then we exhibit two such solutions π^0 and π^1, and we propose hand-tailored cuts to avoid exploring some solutions that can be expressed as the midpoint of π^0 or π^1 and another solution.

Another contribution of this paper is a method that aims at finding lower and upper bounds for the dual values. The additional piece of information needed by this method is the value of a lower bound, which is obtained using a set of precomputed dual solutions. This value permits one to update the lower bounds for several dual values. Then the method exploits the characterization of the maximal solutions to propagate these lower bounds to all dual variables. We also generalize a result that allows us to remove the small item sizes if they cannot change the value of an optimal solution.

Another way of taking into account an interesting (hopefully optimal) known dual solution is to restrict the search space to a small box around this solution, similarly with [3]. To compute these boxes, we use the values given by the known dual solution that yields the best lower bound among a subset of functions. Such a dual solution can be obtained by applying a so-called *dual-feasible function* to the item sizes. The boxes are increased if they are too small, and at the end they are removed and the search is resumed. The idea is to make the method converge faster to an *interesting* area by avoiding large and useless fluctuations of the dual variables.

All the methods we propose lead to dual constraints (additional columns) that are introduced in a column generation based method for the CSP. Computational experiments on a set of hard instances from the literature and randomly generated confirm that each method reduces the number of column generation iterations and the total computing time.

Section 2 reviews several results of the literature concerning dual solutions and dual cuts. In Section 3 we propose new dual cuts. Section 4 is about bounds for the dual values. Section 5 contains numerical

experiments.

2 Definitions and previous results

In this section we briefly review several concepts used in this paper. First we describe the model of Gilmore and Gomory [15, 16] for the CSP. Then we deal with properties of dominant dual solutions of the CSP. Finally we discuss on how *dual cuts* can be applied to speed up column generation processes.

Throughout the remainder, we will use the following notation for the cutting-stock problem (CSP). An instance $D = (C, I, b)$ of CSP is composed of a roll size $C \in \mathbb{N}^*$, a set $I \subseteq \{1, \ldots, C\}$ of item sizes, and a demand function b defined from I to $\mathbb{N}^*$, which associates with each $i \in I$ a demand $b_i \in \mathbb{N}^*$.

2.1 Column generation based model of Gilmore and Gomory [15, 16]

A combination of items of I in a roll is called a *pattern*. Each possible cutting pattern is described by a column $p = (a_{1p}, \ldots, a_{ip}, \ldots, a_{|I|p})^T$, where a_{ip} is the number of items of width i in the pattern p. The model of Gilmore and Gomory [15, 16] is expressed as follows:

$$min \sum_{p \in P} x_p \tag{1}$$

$$\text{subj. to} \quad \sum_{p \in P} a_{ip} x_p \geq b_i, \quad i \in I \tag{2}$$

$$x_p \geq 0, \qquad \forall p \in P \tag{3}$$

$$x_p \text{ integer }, \qquad \forall p \in P \tag{4}$$

where P is the set of valid patterns. A valid cutting pattern is such that

$$\sum_{i \in I} a_{ip} i \leq C, \qquad \forall p \in P \tag{5}$$

$$a_{ip} \geq 0 \text{ and integer}, \quad \forall p \in P, i \in I \tag{6}$$

As the number of possible cutting patterns may be large, the master program is initialized with only a subset of cutting patterns (columns). It finds the best solution that only uses the patterns available. Then an optimization algorithm is executed to find a pattern that would improve the quality of the current solution. This is equivalent to solving a knapsack problem, which can be done using dynamic programming or an enumerative algorithm. The dual of the LP above reads:

$$max \qquad \sum_{i \in I} b_i \pi_i \tag{7}$$

$$\text{subj. to} \quad \sum_{i \in I} a_{ip} \pi_i \leq 1, \forall p \in P \tag{8}$$

$$\pi_i \geq 0 \tag{9}$$

A dual solution π is a vector $(\pi_1, \pi_2, \ldots, \pi_{|I|})$ that obeys all the constraints (8) and (9). We denote the value of the solution yielded by π for a given instance (C, I, b) as $v(\pi)$.

2.2 Superadditive functions, maximal solutions

Throughout the paper, we use the notions of *superadditivity* and *maximality*. We now discuss on their relation with dual solutions.

Definition 1. *A function f is superadditive if $\forall x, y,\ f(x) + f(y) \leq f(x + y)$.*

By abuse of language, we shall say increasing and superadditive dual solutions if their coefficients can be obtained by applying an increasing and superadditive function. A function that leads to a dual solution is said dual-feasible [20] (DFF). DFF are generally defined from $[0,1]$ to $[0,1]$. In the linear programming relaxation of the CSP, a solution maps values of item sizes in $[0,C]$ into dual solutions in $[0,1]$. We could also define a function from $[0,1]$ to $[0,1]$ by scaling the item sizes, *i.e.*, dividing the item sizes by the roll sizes. However, for the sake of simplicity, we will define dual-feasible functions from $[0,C]$ to $[0,1]$, and thus dual values and DFF will be written in the same fashion.

A dual solution π is *dominated* by another solution π' if it cannot lead to a better solution for any data. If a dual solution is not dominated by any other, it is *maximal*. A corresponding concept has been defined for DFF by [8]. The following proposition is a slight rewriting of their result.

Proposition 1. *(Theorem 1. of [8]). A dual-feasible function is* maximal *if and only if* $f(0) = 0$, f *is increasing,* f *is superadditive, and* f *is such that* $\forall i = 1, \ldots, C$, $f(i) + f(C-i) = 1$.

In the following, we will say that if a function is such that $\forall i = 1, \ldots, C$, $f(i) + f(C-i) = 1$, it is *symmetric*.

A dual solution is *maximal* if and only if there is a maximal DFF (MDFF) f such that $f(i) = \pi_i$ for all i of I and all b_i. Nemhauser and Wolsey [27] describe conditions that characterize *dominant* solutions of the knapsack polytope (*i.e.* extreme points of the polytope). Not all maximal solutions are extreme points of the polyhedron. This means that even when we ensure that the considered solutions are maximal, there are many solutions that are not useful. This is confirmed by the experimentations of Carlier and Néron [7, 8] in the context of cumulative scheduling problems.

2.3 Dual cuts

In order to speed up the column generation procedure for solving the CSP, Valério de Carvalho [31] added cuts that exclude solutions that are not superadditive, or not increasing. Only a subset of such cuts were considered, because they are exponential in number. The dual cuts added were the following.

$$\pi_i \leq \pi_j \text{ for } i < j$$

$$\pi_i + \pi_j \leq \pi_{i+j} \text{ for } i, j \in I$$

Dual cuts can be seen as constraints that avoid oscillations of the dual variables. From a primal point of view, they can also be seen as *exchange vectors* (see e.g. [28]), which allow to implicitly generate a whole set of primal columns from the current set of columns available.

2.4 Bounds on the dual values [5]

In the remainder, several results depend on the fact that we know initial lower and upper bounds for the dual values. The following bounds are valid for any maximal dual solution. This result is rewriting of a claim of [5].

$$\begin{aligned} 0 \leq \pi_i \leq \frac{1}{\lfloor C/i \rfloor}, &\quad \text{for } 0 < i < C/2 \\ \pi_{C/2} = 1/2 & \\ 1 - \frac{1}{\lfloor C/(C-i) \rfloor} \leq \pi_i \leq 1, &\quad \text{for } C/2 < i < C \\ \pi_C = 1 & \end{aligned}$$

In the sequel we will use respectively l_i and u_i for a lower bound and an upper bound of the value of π_i.

3 New dual cuts

In this section, we introduce new dual cuts, which can be applied to a linear program for solving the CSP. The first cut we propose is similar to those of [31]: it ensures that the dual solution will be *symmetric*.

The other cuts are based on a different idea, and define a new family of cuts. The motivation is to avoid solutions that are linear combinations of known solutions, even if they are maximal. Whereas the cuts of [31] leave at least one optimal solution, our procedures cut off all dual solutions with a given structure, of which only one needs to be checked in order to see whether the optimal solution is among these.

3.1 A first family of cuts

The dual cuts proposed in [31] exclude dual solutions that are not increasing and superadditive. Following the characterization of [27], we can also cut solutions that are not *symmetric* (Cf. Proposition 1). This would lead to the following family of cuts.

Proposition 2.

$$\pi_i + \pi_{C-i} = 1, \forall i, C - i \in I$$

Proof. If for two values i and $C - i$, $\pi_i + \pi_{C-i} < 1$, then the DFF f that yields the dual solution is not maximal, since it is not symmetric. Consequently, this solution can be safely cut. □

When C is large, there is a small chance that there are two such items of size i and $C-i$ in the instance, even if a suitable preprocessing method is applied. Practically speaking, we use a slight generalization of the cuts above.

Proposition 3. *The following family of cuts is valid.*

$$\pi_i + \pi_j \geq 1, \forall i, j \in I : i + j \geq C \tag{10}$$

Proof. Using Proposition 2, we know that $\pi_i + \pi_{1-i} = 1$. Since dominant dual values follow an increasing rule, $\pi_j \geq \pi_{1-i}$ and thus $\pi_i + \pi_j \geq 1$. □

This family of cuts is straightforward to apply. When combined with the cuts of [31], cuts (10) may lead to improved results.

3.2 A general result

In the remainder of this section, our goal is to exclude dual solutions that are linear combinations of a given dual solution π' and another solution. First we state general results, which are independent of any chosen solution π'. This leads to a general framework for deriving cuts. Then two specific functions π^0 and π^1 are studied, and more specific cuts are derived.

An issue is that such cuts may also cut π'. This means that the value of an optimal solution found when these cuts are applied may be strictly less than the value of an actual optimal solution. If that happens, π' is an optimal solution. The following general proposition ensures that these cuts are safe if π' has been recorded.

Let π' be a dual solution of the CSP and E be a cut that excludes only π' and a set of dual solutions that can be expressed as a convex combination of π' and another dual solution.

Proposition 4. *If OPT is the optimal solution of the LP, $\widehat{OPT}$ the optimal solution of the LP obtained when E is applied, $OPT = max\{v(\pi'), \widehat{OPT}\}$.*

3.2.1 Characterization of dominated solutions

Using the general result of Proposition 4, powerful cuts can be derived if one is able to characterize solutions that are convex combinations of some other solutions. Before showing how that can be done, we first state a result that helps us proving that a solution is the midpoint of two other solutions, and thus cannot lead to a better result.

Lemma 1. *Let f be a maximal dual-feasible function. Function f only leads to solutions π that are dominated if and only if there exists a maximal dual-feasible function g different from f such that*

$$f(x) = 0 \quad \Longrightarrow \quad g(x) = 0 \tag{11}$$

$$f(x) + f(y) = f(x+y) \quad \Longrightarrow \quad g(x) + g(y) = g(x+y) \tag{12}$$

Proof. We first show that if both properties above are satisfied, $h = (f - \epsilon g)/(1-\epsilon)$ is a superadditive and increasing function for a small-enough positive value ϵ.

It is sufficient to show that h is superadditive, and positive (if the two properties are true, the function must be increasing). For the sake of simplicity, we make the proof with $f - \epsilon g$.

The function $f - \epsilon g$ is positive, since by assumption, there is no value x such that $f(x) = 0$ and $g(x) \neq 0$. Since the domain of the function is discrete, one can always find a small enough ϵ to ensure that $f - \epsilon g$ is always positive.

Now we prove that the function is also superadditive. For all x, y such that $f(x) + f(y) < f(x+y)$, one can always find a sufficiently small ϵ to ensure that $(f - \epsilon g)(x) + (f - \epsilon g)(y) \leq (f - \epsilon g)(x+y)$. For x, y such that $f(x) + f(y) = f(x+y)$, the relation is satisfied, since in this case $g(x) + g(y) = g(x+y)$.

We have shown that h is superadditive and increasing if the properties above are satisfied. Function f can be rewritten as $\epsilon g + (1-\epsilon)h$, with g a MDFF, and h a superadditive and increasing function. So any solution π yielded by f can be expressed as the midpoint of two other solutions. Consequently it is dominated.

□

3.2.2 Deriving the cuts

In the following we adapt the conditions of Lemma 1 to dual solutions instead of dual-feasible functions (which is straightforward), and we relax these conditions so they can be expressed as linear constraints. Constraints (13) and (14) below are respectively related to constraints (11) and (12). For condition (14), we consider any pair of values such that $\pi_j + \pi_k < \pi_{j+k}$ in π and compute a lower bound for a weighted sum of the dual values if $\pi_j + \pi_k = \pi_{j+k}$ in a dual solution. This bound is obtained by improving the lower bounds of π_j and π_k, and then by propagating these lower bounds using the superadditivity.

Let π be a maximal dual solution defined for values $\{1, \ldots, C\}$ and i^* be the smallest value smaller than or equal to $C/2$ such that $\pi_{i^*} > 0$ in π. If π is maximal, this value exists since at least $\pi_{C/2} = 1/2$. Let also Δ be the set of pair of values (j,k) such that $\pi_j + \pi_k < \pi_{j+k}$ in π. In the following, we use the values l_i and u_i, the bounds defined in Section 2. Without loss of generality, we consider that $j < k$.

Note that in the following, the values $\widehat{t}_i^{(j,k)}$ have to be computed by increasing value of i, and even for values of $\{1, \ldots, C\}$ that are not included in I.

Proposition 5. *If π does not yield the optimal solution, there exists an optimal solution π' such that either*

$$\pi'_{i^*} = 0 \tag{13}$$

or the following dual cut is valid, for any set of real parameters λ_i, $i \in I$:

$$\sum_{i \in I} \pi'_i \lambda_i \geq \min_{(j,k) \in \Delta} \left\{ l_{j+k}\lambda_{j+k} + \sum_{i \in I \setminus \{j,k\}} \widehat{t}_i^{(j,k)} \lambda_i \right\} \tag{14}$$

with

$$\widehat{t}_i^{(j,k)} = \begin{cases} l_i & \text{if } i < j \\ max\left\{l_i, l_{j+k} - u_j\right\} & \text{if } i = j \\ max\left\{l_i, \underset{l+m=i}{max}\{\widehat{t}_l^{(j,k)} + \widehat{t}_m^{(j,k)}\}\right\} & \text{if } j < i < k \\ max\left\{l_i, \underset{l+m=i}{max}\{\widehat{t}_l^{(j,k)} + \widehat{t}_m^{(j,k)}\}, l_{j+k} - u_k\right\} & \text{if } i = k \\ max\left\{l_i, \underset{l+m=i}{max}\{\widehat{t}_l^{(j,k)} + \widehat{t}_m^{(j,k)}\}\right\} & \text{if } i > k \end{cases}$$

Proof. We have to show that any solution that is cut is also a linear combination of π and another dual solution. It will follow that the cut is safe if π does not yield the optimal solution.

First we show that the value $\widehat{t}_i^{(j,k)}$ is a valid lower bound for the value that π'_i can take when $\pi'_j + \pi'_k = \pi'_{j+k}$. By the initial assumption, $\widehat{t}_i^{(j,k)} \geq l_i$. Since $\pi'_j + \pi'_k = \pi'_{j+k}$, we know that $\pi'_k \geq l_{j+k} - u_j$. For $i = j$, a similar deduction can be made. For any value greater than j, the validity of $\widehat{t}_i^{(j,k)}$ is due to the superadditivity of a MDFF.

This means that the right-hand term of Equation (14) is a lower bound for $\sum_{i \in I} \pi'_i \lambda_i$ when there is at least one pair (j,k) in Δ such that $\pi'_j + \pi'_k = \pi'_{j+k}$. Consequently, in any solution π' that does not respect constraint (14), there is no pair (j,k) of Δ such that $\pi'_j + \pi'_k = \pi'_{j+k}$.

Using Lemma 1, we deduce that if a solution does not respect any of the two constraints (13) and (14), it is a linear combination of π and another solution. Under the initial assumption, any such solution can be safely cut.

□

This leads to a general three-step method for deriving cuts when a dual solution π is known (take one of those surveyed in [9] for example). Since we know that at least one of the two conditions has to hold, it means that either both hold (step 3), or only (13) holds (step 5), or only (14) holds (step 8). The method is described by Algorithm 1, where i^* is defined as above.

Algorithm 1: A generic method for applying dual cuts that exclude a given dual solution

1 Create the original LP ;
2 Apply the cuts (13) and (14) to the LP ;
3 Solve the LP obtained ;
4 Remove the cuts (14) from the LP ;
5 Solve the LP obtained ;
6 Remove the cut (13) from the LP;
7 Add the cut $\pi_i^* > 0$ and the cuts (14) to the LP ;
8 Solve the LP obtained ;
9 Return the best value obtained ;

The quality of the cuts depends on the choice of initial dual solution and on the values λ_i. Several methods can be used to compute a suitable set of coefficients, either by taking into account the demand for each item, or randomly. Practically speaking, we used random coefficients, which led to the best results.

3.3 The solution π^0

Any dual solution can be used in the framework described above. When specific dual solutions are considered, more powerful cuts can be derived. First we study a dual solution with a simple structure.

- $\pi_i = 0$ for $i < C/2$
- $\pi_{C/2} = 1/2$

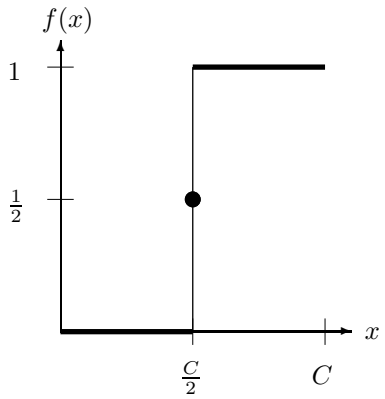

Figure 1: A dual-feasible function that leads to π^0

- $\pi_i = 1$ for $i > C/2$

The value related to this solution is equal to the number of items of size greater than $C/2$ plus the number of items of size $C/2$ divided by two. We name this dual solution π^0. The corresponding dual-feasible function is pictured in Figure 1. We chose this solution for its simple structure, and for it would lead to the most unbalanced solutions, in terms of the values of the dual variables.

3.3.1 Characterization of solutions combined with π^0

Proposition 6. *Let f be a MDFF such that, for all $x < C/2$, $f(x) < x/C$. Then any dual solution yielded by f is the midpoint of π^0 and another dual solution.*

Proof. We use Lemma 1 to show this result. By definition, π^0 can be obtained by the mean of a MDFF, that we name g_0. We have to show that conditions (11) and (12) are verified, where g_0 plays the role of g and f is defined as above.

Since $g_0(x) = 0$ for all $x < C/2$, and since for any MDFF f, $f(x) > 0$ if $x \geq C/2$, condition (11) is verified.

Now we show that condition (12) is also verified, *i.e.* for all pairs x, y such that $f(x)+f(y) = f(x+y)$, we have $g_0(x)+g_0(y) = g_0(x+y)$. The proof consists in six cases, which cover all possibilities. Three are related to cases where $g_0(x) + g_0(y)$ is always equal to $g_0(x + y)$, the others are related to cases where $f(x) + f(y)$ cannot be equal to $f(x + y)$.

In the three following cases, $g_0(x) + g_0(y)$ is always equal to $g_0(x + y)$.

1. If $x + y < C/2$, $g_0(x) + g_0(y) = 0 + 0 = 0$ and $g_0(x + y) = 0$.
2. If $x < C/2$ and $y > C/2$, $g_0(x) + g_0(y) = 0 + 1 = 1$ and $g_0(x + y) = 1$.
3. If $x = y = C/2$, $g_0(x) + g_0(y) = 1/2 + 1/2 = 1$ and $g_0(x + y) = g_0(1) = 1$.

In the two following cases, $f(x) + f(y)$ cannot be equal to $f(x + y)$.

1. If $x < C/2$, $y < C/2$, and $x + y = C/2$, by assumption $f(x) + f(y) < x/C + y/C = 1/2$ and since f is maximal (and then symmetric), $f(x + y) = f(C/2) = 1/2$.
2. If $x < C/2$, $y \leq C/2$, and $x + y > C/2$, $f(x) + f(y) < x/C + y/C$. By symmetry, $f(x + y) = 1-f(C-x-y)$. Since $C-x-y < C/2$, $f(C-x-y) < 1-\frac{x+y}{C}$. Consequently, $f(x+y) > x/C+y/C$.

□

We use Proposition 6 to detect dual solutions that are dominated by π^0 and another solution. The purpose of Proposition 7 is to show a way of verifying that a dual solution is yielded by a dominated MDFF.

Proposition 7. *Let $D = (C, I, b)$ be an instance of CSP and π be a maximal dual solution. If $i < C/2$ implies $\pi_i < i/C$, then there exists a dual solution π' different from π such that $v(\pi) \leq max\{v(\pi^0), v(\pi')\}$.*

Proof. First we write a function $\overline{f}$ that yields π.

$$\overline{f} : i \mapsto \begin{cases} 0 & \text{if } i < min\{I\} \\ \pi_i & \text{if } i \in I \\ max_{j:0<j<i}\{\overline{f}(j) + \overline{f}(i-j)\} & \text{if } i \notin I, min\{I\} < i < C/2 \\ 1/2 & \text{if } i = C/2 \\ 1 - \overline{f}(C-i) & \text{if } i \notin I, i > C/2 \end{cases} \tag{15}$$

Since we consider a maximal dual solution, $\overline{f}$ is maximal by construction. We show that $\overline{f}$ is such that for all $i < C/2$, $\overline{f}(i) < i/C$. For values in I, it is true by assumption. For values i that are not in I, we show the result by recurrence.

By initial assumption, either 1 does not pertain to I, or $\pi_1 < 1/C$. In both cases, $\overline{f}(1) < 1/C$.

Now suppose that for a given value $k < C/2 - 1$, we have $\overline{f}(i) < i/C$ for all values $i < k$. If $k+1$ is in I, $\overline{f}(k+1) < (k+1)/C$. If $k+1$ is not in I, $\overline{f}(k+1)$ is equal to $\overline{f}(j) + \overline{f}(k+1-j)$ for a given j. By assumption, $\overline{f}(j) < j/C$ and $\overline{f}(k+1-j) < (k+1-j)/C$. Thus $\overline{f}(k+1) < (k+1)/C$. Consequently, for all values i such that $i < C/2$, $\overline{f}(i) < i/C$.

Proposition 6 tells us that any dual solution yielded by $\overline{f}$ is dominated by π^0 and another dual solution. Since $\overline{f}(i) = \pi_i$ for all $i \in I$, this concludes the proof. □

3.3.2 Deriving the cuts

Cuts can be obtained using the general result of Proposition 5, but since π^0 has a specific structure, a hand-tailored method can be designed. The idea is to cut some solutions where there are no values $i < C/2$ such that $\pi_i \geq i/C$. This cannot be verified directly by the mean of a linear cut, so we study the different values that can be taken by the sum of the dual values.

There must be one value j such that $\pi_j \geq j/C$. Our idea is to consider the weighted sum of the dual values for each possible j. For this purpose, we compute $\widehat{x}_i^j$, the minimum value that π_i can take when $\pi_j = j/C$. All values π_i with $i < j$ can be equal to 0, whereas the values larger than j have to follow the superadditivity rule.

Note that $\widehat{x}_i^j$ have to be computed by increasing values of i, and even for values that are not in I.

Proposition 8. *If $v(\pi^0) < OPT$, and $I_1 = \{i \in I, i < C/2\}$, for any positive values $\lambda_i, i \in I_1$, the following cut is valid*

$$\sum_{i \in I_1} \pi_i \times \lambda_i \geq \min_{j \in I_1} \left\{ \sum_{i \in I_1} \widehat{x}_i^j \lambda_i \right\} \tag{16}$$

with

$$\widehat{x}_i^j = \begin{cases} 0 & \text{if } i < j \\ j/C & \text{if } i = j \\ \max\limits_{l,m \in I : l+m=i} \{\widehat{x}_l^j + \widehat{x}_m^j\} & \text{if } i > j \end{cases}$$

Proof. Proposition 7 tells us that there must be at least one item $j \in I_1$ such that $\pi_j \geq j/C$. For a given j, $\widehat{x}_i^j$ is the minimum value that π_i can take in a maximal solution when $\pi_j = j/C$ (this bound is obtained by considering the superadditivity only). Consequently $\sum_{i \in I_1} \widehat{x}_i^j \lambda_i$ is a lower bound for the weighted sum of the dual values when $\pi_j = j/C$. If one tests all possibility for j, and keep the minimum sum, a lower bound for $\sum_{i \in I_1} \pi_i \times \lambda_i$ is obtained. □

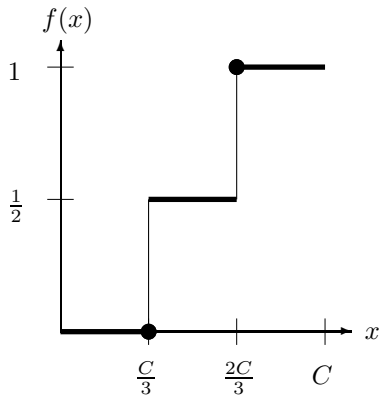

Figure 2: A dual-feasible function that leads to π^1

Note that even if some items sizes do not appear in the instance, they have to be considered in the computation of $\widehat{x}_i^j$. For a given value of C, a given set I, and a given set of parameters, the right-hand part of the inequality can be computed only once for all executions. When such a cut is applied, one has to be aware that solution π^0 is also cut. For the method to remain exact, $v(\pi^0)$ has to be recorded.

As an example, cut (17) is obtained by using (16) with $\lambda_j = 1$ and $\lambda_{j-1} = 1$, and all other $\lambda_k = 0$. An issue is to find an interesting set of parameters λ_i.

Proposition 9. *If π^0 is not an optimal solution, and j is the largest value of I strictly less than $C/2$,*

$$\pi_j + \pi_{j-1} > \frac{j}{C} \tag{17}$$

is a valid dual cut.

Corollary 1. *If π^0 is not the optimal solution, and j is the largest value of I strictly less than $C/2$,*

$$\pi_j \geq \frac{j}{2C} \tag{18}$$

Example 1. *If $C = 100$, and the largest size of a* small *item is* 30*, we have the cut $\pi_{30} \geq 3/20$. Initially the upper bound for π_{30} is $\frac{1}{\lfloor 100/30 \rfloor} = 1/3$, which means that the size of its domain is divided by two.*

3.4 The solution π^1

We now consider the following dual solution, which we name π^1 (see Figure 2 for the corresponding dual-feasible function).

- $\pi_i = 0$ if $i \leq C/3$
- $\pi_i = 1/2$ if $i \in]C/3, 2C/3[$
- $\pi_i = 1$ if $i \geq 2C/3$

In the following, we study this dual solution, and derive dual cuts to exclude solutions that are obtained by linear combination of π^1 and another solution. This is more difficult than with π^0, since its structure is slightly more complex. A three-step method is used, whereas only one step was needed for π^0.

3.4.1 Characterization of solutions combined with π^1

Proposition 10. *Let f be a maximal function. If $x \leq C/3$ implies $f(x) < 3x/4C$, and $x \in]C/3, C/2[$ implies $f(x) \geq 3x/4C$, then any dual solution yielded by f is the midpoint of π^1 and another dual solution.*

Proof. We use Lemma 1 to show this result. By definition, π^1 can be obtained by applying a MDFF to the item sizes, that we name g_1: $g_1(x) = 0$ if $x \leq C/3$, $g_1(x) = 1/2$ if $x \in]C/3, 2C/3[$ and $g_1(x) = 1$ if $x \geq 2C/3$. We show that the two conditions of Lemma 1 are verified, where g_1 plays the role of g and f is defined as above.

We have $g_1(x) = 0$ for $x \leq C/3$. For the values in $]C/3, C/2[$, f is strictly greater than 0 by assumption. Consequently the first condition of Lemma 1 is verified.

Now we show that for each pair x, y such that $f(x) + f(y) = f(x + y)$, $g_1(x) + g_1(y) = g_1(x + y)$. Without loss of generality, we assume that $x \leq y$. Eight cases have to be considered.

In the four following cases, $g_1(x) + g_1(y)$ is always equal to $g_1(x + y)$.

1. If $x \leq C/3$, $y \leq C/3$ and $x + y \leq C/3$, $g_1(x) + g_1(y) = 0 + 0 = 0$ and $g_1(x + y) = 0$.
2. If $x \leq C/3$ and $y \in]C/3, 2C/3[$ and $x + y \in]C/3, 2C/3[$, $g_1(x) + g_1(y) = 0 + 1/2 = 1/2$ and $g_1(x + y) = 1/2$.
3. If $x \in]C/3, 2C/3[$ and $y \in]C/3, 2C/3[$, $g_1(x) + g_1(y) = 1/2 + 1/2 = 1 = g_1(x + y)$.
4. If $x \leq C/3$ and $y \geq 2C/3$ and $x + y \geq 2C/3$, $g_1(x) + g_1(y) = 0 + 1 = 1$ and $g_1(x + y) = 1$.

In the five following cases, $f(x) + f(y)$ cannot be equal to $f(x + y)$.

1. If $x \leq C/3$, $y \leq C/3$ and $x+y \in]C/3, C/2]$, $f(x)+f(y) < 3x/4C+3y/4C$ and $f(x+y) \geq 3(x+y)/4C$.
2. If $x \leq C/3$, $y \leq C/3$ and $x + y \in]C/2, 2C/3[$, $f(x) + f(y) < 3(x + y)/4C$. Since $x + y < C/3$, this is smaller than $3(2C/3)/4C = 1/2$ and $f(x + y) \geq 1/2$ since f is a MDFF.
3. If $x = C/3$, $y = C/3$ and $x + y = 2C/3$, $f(x) + f(y) < 3(x + y)/4C$ and by symmetry $f(x + y) = 1 - f(C - x - y)$. Since $C - x - y \leq C/3$, this is greater than $1 - \frac{3(C-x-y)}{4C} = 1/4 + 3(x + y)/4C$.
4. If $x \leq C/3$, $y \in]C/3, C/2]$ and $x + y \geq 2C/3$, we have $f(y) \leq 1/2$ since $y \leq C/2$ and f is nondecreasing and symmetric. Thus $f(x) + f(y) < 3x/4C + 1/2 \leq \frac{3}{4C}\frac{C}{3} + 1/2 = 3/4$ and by symmetry $f(x+y) = 1-f(C-x-y) > 1-\frac{3(C-x-y)}{4C} = \frac{C+3(x+y)}{4C} = 1/4+\frac{3(x+y)}{4C} \geq 1/4+\frac{3(2C/3)}{4C} = 1/4 + 1/2 = 3/4$.
5. If $x \leq C/3$, $y \in]C/2, 2C/3[$ and $x + y \geq 2C/3$, $f(x) < \frac{3x}{4C}$ and by symmetry $f(y) = 1 - f(C - y)$. Since $C - y \in]C/3, C/2[$, $f(C - y) \geq 3(C - y)/4C$ and thus $f(y) \leq 1 - \frac{3(C-y)}{4C}$. Consequently, $f(x) + f(y) < \frac{3x}{4C} + 1 - \frac{3(C-y)}{4C} = \frac{C+3x+3y}{4C}$ and by symmetry $f(x + y) > 1 - \frac{3(C-x-y)}{4C} = \frac{C+3x+3y}{4C}$.

□

The following proposition shows a way of verifying that a given dual solution is yielded by a dominated MDFF. The proof is avoided, since it is similar to Proposition 7 (constructing the underlying MDFF and applying Proposition 10).

Proposition 11. *Let $D = (C, I, b)$ be an instance of CSP and π be a maximal dual solution. If for all $i \leq C/3$, $\pi_i < 3i/4C$, and for all $i \in]C/3, C/2[$, $\pi_i \geq 3i/4C$, then there exists a dual solution π' different from π such that $v(\pi) \leq max\{v(\pi^1), v(\pi')\}$.*

3.4.2 Deriving the cuts

There is no straightforward way of deriving a cut from this property, since the condition to be verified is $\exists i \leq C/3, \pi_i \geq 3i/4C$ **or** $\exists i \in]C/3, C/2[, \pi_i < 3i/4C$. This means that three non-dominated cases are possible: the first condition is verified, the second condition is verified, or both are verified.

This leads to a three-step method, where each step is related to a possible case. First we assume that there are no values $i \leq C/3$ such that $\pi_i \geq 3i/4C$ (only condition 2 has to be verified). We add the corresponding cuts. When the optimal solution under this first assumption is found, cuts are added to ensure that the second condition is also respected, and the program rerun (conditions 1 and 2 have to be verified). Then the cuts added in the first step are removed, and replaced by a cut that can be derived

from the fact that there is at least one value $i \leq C/3$ such that $\pi_i \geq 3i/4C$ (now only condition 1 has to be verified). The method is summed up in Algorithm 2.

The cuts we derived are based on the following propositions, whose proofs are omitted, since they are similar to Proposition 8.

Note that $\widehat{y}_i^j$ has to be computed by increasing values of i and even for values that are not in I.

Proposition 12. *If the optimal solution is such that there are no items $i \leq C/3$ such that $\pi_i \geq 3i/4C$ in the optimal solution, and if $v(\pi^1) < OPT$, then for any positive values $\lambda_1, \ldots, \lambda_k$, and I_1 defined as above,*

$$\sum_{i \in I_1} \pi_i \times \lambda_i \leq \underset{j \in I \cap]C/3, C/2[}{max} \left\{ \sum_{i \in I_1} \widehat{y}_i^j \lambda_i \right\} \tag{19}$$

is a valid dual cut, where $\widehat{y}_i^j$ is defined as follows:

$$\widehat{y}_i^j = \begin{cases} \frac{3j/4C}{\lfloor j/i \rfloor} & \text{if } i < j \\ 3j/4C & \text{if } i = j \\ 1/\lfloor \frac{C}{i} \rfloor & \text{if } i > j \end{cases}$$

Note that $\widehat{z}_i^j$ has to be computed by increasing values of i and even for values that are not in I.

Proposition 13. *If the optimal solution is such that there is an item i of $I \cap [1, C/3]$ such that $\pi_i \geq 3i/4C$, then for any positive values $\lambda_1, \ldots, \lambda_k$, and I_1 defined as above,*

$$\sum_{i \in I_1} \pi_i \times \lambda_i \geq \underset{j \in I \cap [1, C/3]}{min} \left\{ \sum_{i \in I_1} \widehat{z}_i^j \lambda_i \right\} \tag{20}$$

is a valid dual cut, where $\widehat{z}_i^j$ is defined as follows:

$$\widehat{z}_i^j = \begin{cases} 0 & \text{if } i < j \\ 3j/4C & \text{if } i = j \\ \underset{l+m=i}{max} \{\widehat{z}_l^j + \widehat{z}_m^j\} & \text{if } i > j \end{cases}$$

4 Bounding the dual variables

Another way of taking into account the data is to consider an estimation for the number of stock rolls needed. In this section, we propose to use this information to tighten the bounds u_i and l_i for each dual variable π_i. We also describe how variables related to small values can be set to zero. Finally we show that limiting the search space to the neighborhood of an initial *good* dual solution improves previous results.

For the sake of simplicity, **we consider in the remainder that** $I = \{1, \ldots, C\}$. When an item size i is not in the original problem, we consider that $b_i = 0$. In the sequel, $OPT(I)$ is the optimal value of a solution for the set I, LB is a lower bound for $OPT(I)$ and UB an upper bound for this value.

4.1 New bounds

The first method is a lifting procedure for the lower bounds l_i associated with the dual values π_i. It is based on a trial-and-error procedure. A solution is chosen by setting the dual variable π_i to its minimum l_i and all other dual variables to their maximum u_i. If the value of (possibly invalid) solution is smaller than a known lower bound, l_i is not valid, and it can be updated. This reasoning may lead to better results if the superadditivity constraints are considered.

Let $\widehat{l}_i^{(j,\alpha)}$ be a lower bound for the value π_i when $\pi_j = \alpha$. It is defined as follows, similarly to $\widehat{x}_i^j$ of Proposition 8:

Algorithm 2: Applying the cuts (19) and (20)

Compute $v(\pi^1)$;
Add the cut $\sum_{i\in I}\pi_i b_i > v(\pi^1)$;
// Assume that for all $i \leq C/3$, $\pi_i < 3i/4C$ *and there is a* $C/2 > j > C/3$ *such that* $\pi_j < 3j/4C$
Add the cuts $\pi_i < 3i/4C, \forall i \leq C/3$;
Add the cuts (19) ;
Run the column-generation procedure ;
// Assume that there is an $i \leq C/3$, *such that* $\pi_i \geq 3i/4C$ *and a* $C/2 > j > C/3$ *such that* $\pi_j < 3j/4C$
Remove the cuts on the $i \leq C/3$;
Add the cuts (20) ;
Resume the column-generation method ;
// Assume that there is an $i \leq C/3$, *such that* $\pi_i \geq 3i/4C$ *and for all* $C/2 > i > C/3$, $\pi_i \geq 3i/4C$
Remove the cuts (19) ;
Add the cut $\pi_i \geq 3i/4C, \forall C/3 < i < C/2$;
Resume the column-generation method ;
// Let OPT be the optimal value obtained
return $max\{v(\pi^1), OPT\}$;

$$\widehat{l}_i^{(j,\alpha)} = \begin{cases} l_i & \text{if } i < j \\ \alpha & \text{if } i = j \\ max\{l_i, \underset{j+k=i}{max}\{\widehat{l}_j^{(j,\alpha)} + \widehat{l}_k^{(j,\alpha)}\}\} & \text{if } j < i < C - j \\ 1 - \alpha & \text{if } i = C - j \\ max\{l_i, 1 - \frac{\alpha}{\lfloor \frac{j}{C-i} \rfloor}, \underset{j+k=i}{max}\{\widehat{l}_j^{(j,\alpha)} + \widehat{l}_k^{(j,\alpha)}\}\} & \text{if } C - j < i \end{cases}$$

For the sake of simplicity, we also introduce an additional notation $\widehat{u}_i^{(j,\alpha)} = 1 - \widehat{l}_{C-i}^{(j,\alpha)}$, which is the equivalent upper bound obtained by symmetry. Now we define $\theta^j(\alpha)$, a function that associates with the value α the corresponding upper bound on the value of a solution obtained when $\pi_j = \alpha$. This bound is based on $\widehat{l}_i^{(j,\alpha)}$ described above, and on the symmetry of a MDFF.

$$\theta^j(\alpha) = \sum_{i=1}^{C/2} max\{b_i \times \widehat{u}_i^{(j,\alpha)} + b_{C-i} \times \widehat{l}_{C-i}^{(j,\alpha)}, b_i \times \widehat{l}_i^{(j,\alpha)} + b_{C-i} \times \widehat{u}_{C-i}^{(j,\alpha)}\} + b_C \tag{21}$$

Lemma 2. *For* $j < C/2$, $\pi_j = \alpha$ *implies* $\sum_{i\in I}\pi_i b_i \leq \theta^j(\alpha)$.

Proof. We prove this result in two steps. First we prove that $\widehat{l}_i^{(j,\alpha)}$ is a lower bound for π_i when $\pi_j = \alpha$. Then we will deduce that the inequality is valid.

For any value $i \leq j$, $\widehat{l}_i^{(j,\alpha)}$ is a valid lower bound by assumption. If $i = C - j$, we have $\pi_i = 1 - \alpha$ by symmetry. For any value $i > j$, by superadditivity, we have $\widehat{l}_i^{(j,\alpha)} \geq \underset{j+k=i}{max}\{\widehat{l}_j^{(j,\alpha)} + \widehat{l}_k^{(j,\alpha)}\}$. For $C - j < i$, we have $\pi_{C-i} \leq \frac{\alpha}{\lfloor \frac{j}{C-i} \rfloor}$, otherwise π_j could not be equal to α if the coefficients follow a superadditive rule. By symmetry, $\pi_i \geq 1 - \frac{\alpha}{\lfloor \frac{j}{C-i} \rfloor}$.

Now we prove that the inequality is valid. Consider each pair of values i and $C - i$: these values are such that $\pi_i + \pi_{C-i} = 1$. This means that for all i, $\pi_i b_i + \pi_{C-i} b_{C-i} \leq max\{b_i \times \widehat{u}_i^{(j,\alpha)} + b_{C-i} \times \widehat{l}_{C-i}^{(j,\alpha)}, b_i \times \widehat{l}_i^{(j,\alpha)} + b_{C-i} \times \widehat{u}_{C-i}^{(j,\alpha)}\}$. Indeed, either b_i is greater than b_{C-i} and maximizing the expression consists in setting π_i to its upper bound, or in the other case, π_i have to be set to its lower bound.

The validity of the lemma follows.

□

Lemma 2 leads to the following cut, which can be applied when lower bounds l_i and upper bounds u_i are known for the values π_i.

Proposition 14. *Let LB be a valid lower bound for the CSP. The two following constraints are valid dual cuts for the CSP for each $j < C/2$.*

$$\pi_j \leq max\{\alpha : \alpha \in [l_i, u_i], \theta^j(\alpha) \geq LB\}$$

$$\pi_j \geq min\{\alpha : \alpha \in [l_i, u_i], \theta^j(\alpha) \geq LB\}$$

are valid dual cuts for the CSP.

4.2 Computational issues

We now discuss computational issues related to Proposition 14. Function θ depends on the demand for each item type: it may not be convex, and thus the dichotomy cannot be used to determine the extrema. We describe two different approaches. One is based on a relaxation, the other is based on a small linear program.

4.2.1 Solving a relaxation

First we relax the relations in order to obtain a new function $\bar{\theta}^j(\alpha)$ that is decreasing in α. The maximum value of α such that $\bar{\theta}^j(\alpha) \geq LB$ can be computed by dichotomy. Consequently, replacing θ by $\bar{\theta}$ allows us to compute an upper bound for π_j.

The relaxation is obtained as follows: we use the initial upper bounds for the items of size less than j. In this case, only the superadditivity is used to increase the value of lower bounds. Moreover, we modify the upper bound of π_j only if $b_j < b_{C-j}$. For the values such that $b_j \geq b_{C-j}$, the expression is constant. For the values such that $b_j < b_{C-j}$, increasing the value of α decreases the value of $b_i \times \widehat{u}_i^{(j,\alpha)} + b_{C-i} \times \widehat{l}_{C-i}^{(j,\alpha)}$. Consequently the obtained function is decreasing in α.

In order to obtain a lower bound for π_j, the opposite relaxation is applied. We take into account the constraint related to the items less than or equal to j if $b_j > b_{C-j}$. The obtained function is increasing, and thus an upper bound can be computed for π_j by dichotomy.

4.2.2 Solving a linear program

Another way of computing a lower bound for α is to solve the following linear program (22)-(28). In this program, the lower bound is LB, l_i and u_i are respectively a constant lower and a constant upper bound for π_i, and b_i is the demand for item i. Two versions of this program can be used, depending on the fact that I contains all sizes, or only sizes with a non-zero demand.

$$min\ \pi_j \tag{22}$$

subj.to

$$\sum_{i \in I} \pi_i b_i \geq LB \tag{23}$$

$$\pi_{i+k} - \pi_i - \pi_k \geq 0 \text{ for } i, k, i+k \in I \tag{24}$$

$$\pi_i - \pi_k \geq 0, \text{ for } i > k \tag{25}$$

$$\pi_i + \pi_{C-i} = 1 \text{ for } i \in I \tag{26}$$

$$\pi_i \geq l_i \text{ for } i \in I \tag{27}$$

$$\pi_i \leq u_i \text{ for } i \in I \tag{28}$$

An upper bound is obtained by replacing the min by a max.

Discussion 1. *If all item sizes are used, the problem solved is close to the model of Dyckoff [13] for solving the CSP. This means that the computational effort could be even larger than the one needed for the column-generation. When we only consider item sizes of demand strictly greater than 0, it is slightly greater than the initial solution in the column-generation based method of [31]. One way of improving the bound is to add additional rows related to patterns. If too many rows are added, the method could be too much time consuming, so a tradeoff has to be found. For example, in our computational experiments, we already get good results using only a subset of the constraints of (22)-(28).*

Thus a lower bound for the problem can lead to tighter lower and upper bounds for some π_i. Note that increasing a value l_i may help tightening other bounds when it is used in a new definition of $\widehat{l}_i^{(j,\alpha)}$. So the process can be repeated while a value has been modified in the previous step.

This method can also be used with an upper bound UB. In this case, we assume that UB is equal to the optimal value, and we apply the method. If the program leads to a value less than UB, the assumption is false, and UB can be decremented.

4.3 Removing non-useful items

Another way of taking into account the value of the solution is to deduce a set of items whose dual value is zero. For example, if the solution has a value greater than the continuous lower bound, the items of size one have their dual value equal to zero. Symmetrically, when the optimal solution is equal to the number of large items, all dual values related to items smaller than $C/2$ are equal to zero. These are the two extreme cases, which are generalized by Proposition 15 (they are respectively related to cases $k = 0$ and $k = C/2 - 1$).

Proposition 15. *If* $\lceil \frac{\sum_{i\in I} b_i \times i}{C-k} \rceil + 1$ *is a valid lower bound for the number of bins needed to pack the items, then* $OPT(I) = OPT(I \setminus \{1, \ldots, k+1\})$.

Proof. We show that $OPT(I) > OPT(I \setminus \{1, \ldots, k+1\})$ implies $OPT(I) < \lceil \frac{\sum_{i\in I} b_i \times i}{C-k} \rceil + 1$.

Consider any optimal solution π for $I \setminus \{1, \ldots, k+1\}$. From this solution, we will construct a heuristic solution for I, and show that it uses less than $\lceil \frac{\sum_{i\in I} b_i \times i}{C-k} \rceil$ bins. For this purpose we use the following algorithm.

For each item size $i \leq k+1$, create b_i items of size i and put them in a list L. Then pack these items in the bins B_j of π, which are already filled with items of size greater than $k+1$. The rule used is the following: while there are at least $k+1$ units of free space in B_j, pack the next item a of L (in any order) in B_j. Note that item a is always small enough to be packed in the current bin.

By assumption, there are remaining items to pack when all bins have been considered (otherwise $OPT(I) = OPT(I \setminus \{1, \ldots, k+1\})$). Create an infinite number of empty bins and repeat the previous step with these new bins until all items have been packed.

A heuristic solution for I has been constructed. Let δ be the quantity of lost space in this solution and z the number of bins used. We have $z = \frac{\delta + \sum_{i\in I} b_i \times i}{C}$. By construction, all bins have at most k units of waste, except the last bin, which can contain $C-1$ units of waste. Thus $\delta < (z-1) \times k + C$. Consequently, we can write the following inequality: $z < \frac{(z-1)\times k + C + \sum_{i\in I} b_i \times i}{C}$. This leads to $Cz < kz - k + C + \sum_{i\in I} b_i \times i$ and thus $z < \frac{\sum_{i\in I} b_i \times i + C - k}{C-k} \leq \lceil \frac{\sum_{i\in I} b_i \times i}{C-k} \rceil + 1$.

Consequently this solution uses a quantity of bins strictly less than $\lceil \frac{\sum_{i\in I} b_i \times i}{C-k} \rceil + 1$, which is the result sought.

□

4.4 Original trust region

In [3], the authors show that enforcing the dual values to be near *a priori* known optimal dual values (using boxes around the optimal dual values) fasten the convergence. We propose to extend this method when a *good* (possibly non optimal) dual solution is known. Several families of DFF have been proposed in the literature (see [9]). There is a chance that some optimal dual values are *near* the values given by the best DFF used. We propose the following algorithm: we apply a set of DFF, and keep the one which

leads to the best result, then we assume that the dual values are optimal, and add the corresponding boxes around the dual values and solve the LP. Then, these constraints are removed and the search resumed. Algorithm 3 describes our method.

Algorithm 3: Forcing an original trust region

1 consider a given CSP instance D ;
2 find the best DFF f available for D ;
3 build a classical Gilmore-Gomory LP for the CSP, with additional constraints ensuring that dual values are near those given by f ;
4 solve the LP ;
5 if the constraints related to f are too strong, and this step has not been repeated too many times then relax the constraints and goto 4;
6 remove the constraints related to f ;
7 resume the resolution of the LP ;

In practice, we enforce box constraints that are always centered around the values given by the best DFF. If these boxes are too small, their sizes are increased a very small number of times, and the search is resumed.

5 Computational Experiments

To evaluate the strength of the new cuts and methods proposed above, we conducted a set of computational experiments on a set of randomly generated cutting stock instances and on a set of hard instances of the literature [30]. Our focus is on hard instances, *i.e.* those which are solved in a large amount of time using the standard column generation algorithm. We used instances with a large number of different items, and such that the sizes of the items are small compared to the length of the rolls.

Table 1 illustrates the main characteristics of the instances. Column SET identifies the instance set, C stands for the length of the rolls, MIN and MAX correspond to the size of the smallest and the largest item, respectively, NIT is the average number of different item sizes in the instance and B is the average demand per item size. Each set is composed of 10 different instances. The last set is taken from the literature [30], and it is also composed of 10 instances.

SET	C	MIN	MAX	NIT	B
1	100000	1	50000	200	20
2	100000	1	50000	500	20
3	100000	1	35000	200	20
4	100000	1	35000	500	20
5	100000	1	20000	200	20
6	100000	1	20000	500	20
7	100000	1	10000	200	10
8	100000	1	10000	500	10
9	100000	35000	50000	1000	20
HARD	100000	20000	35000	200	20

Table 1: Set of instances

The algorithms were coded in C++, and the CPLEX 10.2 Callable Library [19] was used for some of the optimization subroutines. The tests were performed on a PC with an 2.20 GHz Intel Core Duo processor, and 2GB of RAM.

Clearly, given the length of the rolls and the relative size of the items, solving the knapsack pricing subproblems using dynamic programming is not computationally viable. Hence, we solved them using the branch-and-bound algorithm MT1 for knapsack problems, which is due to Martello and Toth [25].

In the following tables, we compare the main stabilization strategies described in this paper with the standard column generation algorithm with and without the dual cuts proposed in [31]. We show how our strategies make column generation converge faster when they are used alone, and together with the dual cuts of Valério de Carvalho [31]. The cuts are used according to the method of [31]. The first restricted master problem is initialized with a single (artificial) column, and the dual cuts are added prior the resolution of the first restricted master problem. Whenever we considered the specific dual cuts of Valério de Carvalho, we only applied cuts of Type I and II [31].

Five new methods were compared to the standard column generation algorithm, and to the stabilization procedure proposed in [31], namely the dual cuts (16), the algorithms 1 and 2, and the methods based on the computation of dual bounds and on a trust region.

The parameters λ_i in the dual cuts (16), and in the algorithms 1 and 2 were chosen in the following way: the λ_i's were set equal to 1 for the first n items in the respective items set while the remaining λ_i's were set equal to 0; the process was repeated for the $n+1$ first items, and so on, until all the items were finally considered. Thus, the number of cuts that were generated this way is equal to the cardinality of the items set considered in the respective cut. Other experiments were conducted considering other sets of parameters, but none gave better results.

Concerning the implementation of the trust region method, in these experiments, we limited the number of iterations to 10, and we used two dual-feasible functions ([14],[6]) to compute the initial trust region. Initially, the dual variables are forced to be within a box of size proportional to i/C and centered around the values given by the best dual-feasible function. Whenever a dual variable lies at the frontier of that boxes, the size of the box is simply increased.

The results of the different experiments are given in Table 2 through Table 6. These results correspond to the resolution of the linear programming relaxation of the column generation model for the CSP. The execution was stopped after 900 seconds. An instance was considered as *solved* if the optimum was reached within this time limit. The entries in the tables have the following meaning:

- Strategy: solution method used, for example
  - *standard* denotes the standard column generation algorithm,
  - *with dual cuts [31]* means that we added a polynomial set of dual cuts as in [31] before solving the first restricted master,
  - *with dual cuts (16)*) means that we used only the described cuts,
  - *Algorithm 1* means that we used this algorithm with no other dual cuts.
- *solved*: number of instances solved within the time limit,
- *itr*: average number of pricing subproblems solved (column generation iterations);
- *%red. itr*: relative reduction in the number of column generation iterations (compared to the standard algorithm);
- t_{tot}: average solution time (in seconds);
- *%red.* t_{tot}: relative reduction in the total solution time (compared to the standard algorithm).

When a method fails to solve a single instance in a set, the respective rows are filled with entries $-$.

Our experiments show a clear improvement on convergence when compared to standard column generation and to the method proposed in [31]. In fact, the latter seems to be quite ineffective on these hard instances. In many cases, the convergence of column generation greatly deteriorates when these particular cuts are used, and, even if the number of iterations is usually smaller when these cuts are applied alone, the computing time needed to solve the instances is almost always worse.

In general, the number of instances solved to optimality within the time limit increases significantly when the stabilization strategies described in this paper are considered. In terms of the computing time required to solve the instances, the improvement is also impressive. In some cases, it goes up to almost 98%. Furthermore, the reduction is regular. For all the instance sets, there is always more than one strategy that improve significantly the results. In the other cases, a better parametrization of the

methods would certainly give better results. For example, in the instance set 9, the method based on the computation of dual bounds reduced drastically the number of column generation iterations, but the total computing time increased. This is due to the fact that we derived these bounds for each item size. Only for very demanding instances should such an approach be used.

Set	Strategy	*solved*	*itr*	%*red. itr*	t_{tot}	%*red.* t_{tot}
1	standard	10	1112.50		74.71	
	with dual cuts [31]	10	969.00	12.90	90.49	-21.12
	with dual cuts (16)	10	1059.60	4.76	68.21	8.69
	with dual cuts (16) and [31]	10	946.10	14.96	78.79	-5.47
	Algorithm 1	10	1064.80	4.29	73.18	2.04
	Algorithm 1 and dual cuts [31]	10	968.00	12.99	73.31	1.87
	Algorithm 2	10	858.30	22.85	71.02	4.93
	Algorithm 2 and dual cuts [31]	10	768.40	30.93	71.66	4.08
	with dual bounds	10	1059.90	4.73	69.37	7.14
	with dual bounds and cuts [31]	10	946.90	14.89	67.73	9.34
	trust region	10	651.40	41.45	59.95	19.75
	trust region with dual cuts [31]	10	610.90	45.09	96.13	-28.68
2	standard	8	2431.50		654.73	
	with dual cuts [31]	0	–	–	–	–
	with dual cuts (16)	10	2410.00	0.88	522.19	20.24
	with dual cuts (16) and [31]	8	1925.30	20.82	647.05	1.17
	Algorithm 1	10	2367.00	2.65	518.57	20.80
	Algorithm 1 and dual cuts [31]	10	1965.20	19.18	539.92	17.54
	Algorithm 2	10	1759.30	27.65	364.79	44.28
	Algorithm 2 and dual cuts [31]	10	1464.70	39.76	419.07	35.99
	with dual bounds	9	2336.10	3.92	554.66	15.28
	with dual bounds and cuts [31]	8	1913.40	21.31	503.08	23.16
	trust region	10	1494.70	38.53	291.10	55.54
	trust region with dual cuts [31]	10	1260.80	48.15	416.35	36.41

Table 2: Comparing the stabilization strategies on instance sets 1 and 2

6 Conclusion

It is well known that the linear bounds provided by column generation models for the CSP are strong, and hence any method that improves the efficiency of these methods is of great interest. Among the strategies proposed in the literature, dual cuts are among the most promising. In this paper, we proposed new procedures to derive dual cuts for the CSP. With the cuts generated using our approaches, we were able to reduce the number of column generation iterations necessary to solve this problem. We also described new methods for bounding the dual variables, permanently, or during an initialization phase. These methods were tested on a broad range of instances from the literature, and they proved to be effective in many cases.

Many families of dual-feasible functions have been proposed in the literature, each is related to a family of dual solutions. Finding properties of functions that are linear combination of these known dual functions would be a way of improving our results. Applications of these ideas to enumerative methods is a work in progress.

7 Acknowledgements

This work was supported by the Portuguese Science and Technology Foundation through the postdoctoral grant SFRH/BPD/24139/2005 for François Clautiaux and the research project POS_C / 57203 / EIA /

Set	Strategy	*solved*	*itr*	%*red. itr*	t_{tot}	%*red.* t_{tot}
3	standard	9	861.20		218.40	
	with dual cuts [31]	8	701.80	18.51	354.31	-62.23
	with dual cuts (16)	10	791.70	8.07	119.37	45.35
	with dual cuts (16) and [31]	10	703.90	18.27	129.87	40.54
	Algorithm 1	10	797.00	7.45	106.41	51.28
	Algorithm 1 and dual cuts [31]	10	710.60	17.49	137.00	37.27
	Algorithm 2	10	828.20	3.83	193.46	11.42
	Algorithm 2 and dual cuts [31]	9	678.60	21.20	243.57	-11.52
	with dual bounds	10	785.90	8.74	108.35	50.39
	with dual bounds and cuts [31]	10	708.30	17.75	169.02	22.61
	trust region	10	567.60	34.09	82.74	62.12
	trust region with dual cuts [31]	10	499.90	41.95	80.44	63.17
4	standard	5	1545.60		766.15	
	with dual cuts [31]	0	–	–	–	–
	with dual cuts (16)	8	1762.00	-14.00	652.35	14.85
	with dual cuts (16) and [31]	8	1499.20	3.00	712.66	6.98
	Algorithm 1	9	1809.50	-17.07	690.94	9.82
	Algorithm 1 and dual cuts [31]	8	1476.80	4.45	603.15	21.28
	Algorithm 2	8	1857.70	-20.19	705.34	7.94
	Algorithm 2 and dual cuts [31]	6	1247.60	19.28	772.65	-0.85
	with dual bounds	8	1784.20	-15.44	654.16	14.62
	with dual bounds and cuts [31]	6	1227.60	20.57	713.16	6.92
	trust region	10	1297.30	16.06	536.25	30.01
	trust region with dual cuts [31]	9	1107.40	28.35	531.12	30.68

Table 3: Comparing the stabilization strategies on instance sets 3 and 4

2004 for Cláudio Alves and José Valério de Carvalho.

The authors would like to thank the anonymous referees for their constructive comments, which helped improving the quality of the paper.

References

[1] C. Alves and J. Valério de Carvalho. Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, 183(3):1333–1352, 2007.

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

[3] H. Ben Amor, J. Desrosiers, and F. Soumis. Recovering an optimal lp basis from an optimal dual solution. *Operations Research Letters*, 34:569–576, 2006.

[4] H. Ben Amor, J. Desrosiers, and J. Valério de Carvalho. Dual-optimal inequalities for stabilized column generation. *Operations Research*, 54(3):454–463, 2006.

[5] A. Caprara and M. Monaci. Bidimensional packing by bilinear programming. *Mathematical Programming (in press)*, 2007.

[6] J. Carlier, F. Clautiaux, and A. Moukrim. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. *Computers and Operations Research*, 34(8):2223–2250, 2007.

[7] J. Carlier and E. Néron. A new LP-based lower bound for the cumulative scheduling problem. *European Journal of Operational Research*, 127:363–382, 2000.

Set	Strategy	*solved*	*itr*	%*red. itr*	t_{tot}	%*red.* t_{tot}
5	standard	10	658.70		318.76	
	with dual cuts [31]	4	300.50	54.38	803.11	-151.95
	with dual cuts (16)	10	543.80	17.44	168.24	47.22
	with dual cuts (16) and [31]	10	498.70	24.29	154.96	51.39
	Algorithm 1	10	541.90	17.73	154.22	51.62
	Algorithm 1 and dual cuts [31]	10	501.80	23.82	185.80	41.71
	Algorithm 2	9	638.90	3.01	259.33	18.64
	Algorithm 2 and dual cuts [31]	4	325.70	50.55	752.45	-136.05
	with dual bounds	10	552.60	16.11	156.76	50.82
	with dual bounds and cuts [31]	10	498.80	24.28	185.14	41.92
	trust region	9	463.20	29.68	216.51	32.08
	trust region with dual cuts [31]	10	423.70	35.68	110.08	65.47
6	standard	1	591.20		893.11	
	with dual cuts [31]	0	–	–	–	–
	with dual cuts (16)	4	1049.70	-77.55	835.22	6.48
	with dual cuts (16) and [31]	6	978.20	-65.46	769.49	13.84
	Algorithm 1	7	1215.50	-105.60	779.52	12.72
	Algorithm 1 and dual cuts [31]	7	983.00	-66.27	776.13	13.10
	Algorithm 2	0	–	–	–	–
	Algorithm 2 and dual cuts [31]	0	–	–	–	–
	with dual bounds	4	1092.70	-84.83	843.75	5.53
	with dual bounds and cuts [31]	6	1061.20	-79.50	780.78	12.58
	trust region	8	1076.40	-82.07	631.65	29.27
	trust region with dual cuts [31]	4	768.10	-29.92	759.27	14.99

Table 4: Comparing the stabilization strategies on instance sets 5 and 6

[8] J. Carlier and E. Néron. Computing redundant resources for cumulative scheduling problems. *European Journal of Operational Research*, 176(3):1452–1463, 2007.

[9] F. Clautiaux, C. Alves, and J.M Valério de Carvalho. A survey of dual-feasible and superadditive functions. *submitted to Annals of Operations Research*, 2008.

[10] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[11] Z. Degraeve and M. Peeters. Optimal integer solutions to industrial cutting-stock problems: Part 2, benchmark results. *INFORMS Journal on Computing*, 15(1):58–81, 2003.

[12] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

[13] H. Dyckhoff. A new linear programming approach to the cutting stock problem. *Operations Research*, 29:145–159, 1981.

[14] S. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91:11–31, 2001.

[15] P. Gilmore and R. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[16] P. Gilmore and R. Gomory. A linear programming approach to the cutting stock problem - part II. *Operations Research*, 11:863–888, 1963.

[17] J. Goffin and J. Vial. Cutting planes and column generation techniques with the projective algorithm. *Journal of Optimization Theory and Applications*, 65:409–429, 1989.

Set	Strategy	*solved*	*itr*	%*red. itr*	t_{tot}	%*red.* t_{tot}
7	standard	3	786.10		798.85	
	with dual cuts [31]	0	–	–	–	–
	with dual cuts (16)	9	1107.20	-40.85	427.94	46.43
	with dual cuts (16) and [31]	5	1052.50	-33.89	620.80	22.29
	Algorithm 1	9	1127.10	-43.38	362.28	54.65
	Algorithm 1 and dual cuts [31]	9	1044.80	-32.91	352.89	55.83
	Algorithm 2	3	798.40	-1.56	787.24	1.45
	Algorithm 2 and dual cuts [31]	0	–	–	–	–
	with dual bounds	10	1153.60	-46.75	332.40	58.39
	with dual bounds and cuts [31]	4	1023.80	-30.24	652.26	18.35
	trust region	8	1006.90	-28.09	355.22	55.53
	trust region with dual cuts [31]	0	–	–	–	–
8	standard	2	239.60		731.76	
	with dual cuts [31]	3	172.40	28.05	733.71	-0.27
	with dual cuts (16)	10	487.20	-103.34	30.16	95.88
	with dual cuts (16) and [31]	9	433.20	-80.80	137.16	81.26
	Algorithm 1	10	484.50	-102.21	27.74	96.21
	Algorithm 1 and dual cuts [31]	9	431.40	-80.05	113.71	84.46
	Algorithm 2	3	300.50	-25.42	657.27	10.18
	Algorithm 2 and dual cuts [31]	2	170.50	28.84	770.90	-5.35
	with dual bounds	10	484.50	-102.21	24.81	96.61
	with dual bounds and cuts [31]	9	438.10	-82.85	120.59	83.52
	trust region	9	452.90	-89.02	107.07	85.37
	trust region with dual cuts [31]	4	337.00	-40.65	557.15	23.86

Table 5: Comparing the stabilization strategies on instance sets 7 and 8

[18] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. A Series of Comprehensive Studies in Mathematics. Springer-Verlag, 1993.

[19] Ilog. *Ilog CPLEX 10.0 Reference Manual*. 9, rue de Verdun, BP 85, F-92453, Gentilly, France, 2006.

[20] D. S. Johnson. Near optimal bin packing algorithms, 1973. Dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts.

[21] B. Kallehauge, J. Larsen, and O. Madsen. Lagrangean duality applied on vehicle routing with time windows. *Computers and Operations Research*, 33(5):1464–1487, 2006.

[22] S. Kim, K. Chang, and J. Lee. A descent method with linear programming subproblems for nondifferentiable convex optimization. *Mathematical Programming*, 71:17–28, 1995.

[23] M. Luebbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[24] R. Marsten, W. Hogan, and J. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

[25] S. Martello and P. Toth. *Knapsack problems - Algorithms and Computer Implementation*. Wiley, Chichester, 1990.

[26] P. Neame. *Nonsmooth Dual Methods in Integer Programming*. PhD thesis, University of Melbourne, Australia, 1999.

[27] G. L. Nemhauser and L.A. Wolsey. Integer and combinatorial optimization, 1998.

Set	Strategy	*solved*	*itr*	%*red. itr*	t_{tot}	%*red.* t_{tot}
9	standard	10	989.50		22.59	
	with dual cuts [31]	10	921.10	6.91	74.33	-229.11
	with dual cuts (16)	10	2.00	99.80	0.36	98.41
	with dual cuts (16) and [31]	10	2.00	99.80	0.31	98.64
	Algorithm 1	10	2.00	99.80	0.47	97.91
	Algorithm 1 and dual cuts [31]	10	2.00	99.80	0.39	98.26
	Algorithm 2	10	979.80	0.98	25.07	-11.01
	Algorithm 2 and dual cuts [31]	10	923.90	6.63	64.58	-185.94
	with dual bounds	10	1.00	99.90	31.16	-37.95
	with dual bounds and cuts [31]	10	1.00	99.90	32.76	-45.02
	trust region	10	9.00	99.09	0.70	96.91
	trust region with dual cuts [31]	10	9.00	99.09	5.04	77.69
HARD	standard	10	751.10		32.28	
	with dual cuts [31]	10	438.10	41.67	30.55	5.37
	with dual cuts (16)	10	699.60	6.86	25.91	19.74
	with dual cuts (16) and [31]	10	433.70	42.26	25.15	22.09
	Algorithm 1	10	705.40	6.08	25.38	21.38
	Algorithm 1 and dual cuts [31]	10	433.40	42.30	24.55	23.93
	Algorithm 2	10	760.60	-1.26	25.51	20.98
	Algorithm 2 and dual cuts [31]	10	476.90	36.51	25.33	21.52
	with dual bounds	10	664.70	11.50	25.21	21.91
	with dual bounds and cuts [31]	10	435.90	41.97	23.65	26.74
	trust region	10	369.10	50.86	10.99	65.96
	trust region with dual cuts [31]	10	309.20	58.83	11.12	65.56

Table 6: Comparing the stabilization strategies on instance sets 9 and HARD

[28] N. Perrot. *Integer Programming Column Generation Strategies for the Cutting Stock Problem and its Variants*. PhD thesis, Université Bordeaux 1, 2005.

[29] J. Puchinger and G. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2005.

[30] A. Scholl, R. Klein, and C. Jurgens. BISON: a fast hybrid procedure for exactly solving the one-dimensional bin-packing problem. *Computers and Operations Research*, 24:627–645, 1997.

[31] J.M. Valério de Carvalho. Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005.

[32] G. Wäscher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130, 2007.