Convergent lower bounds for packing problems via restricted dual polytopes

Daniel Cosmin Porumbel, François Clautiaux

Convergent Lower Bounds for Packing Problems via Restricted Dual Polytopes

Daniel Cosmin Porumbel and François Clautiaux

Université d'Artois, LGI2A
Université de Lille 1, LIFL UMR 8022, INRIA Lille Nord Europe
daniel.porumbel@univ-artois.fr,francois.clautiaux@univ-lille1.fr

Abstract. Cutting-stock and bin-packing problems have been widely studied in the operations research literature for their large range of industrial applications. Many integer programming models have been proposed for them. The most famous is from Gilmore and Gomory [5] and relies on a column generation scheme. Column generation methods are known to have convergence issues, and (when minimizing) no useful lower bounds are produced before a possibly large number of iterations. We propose a new approach that converges to the optimum through a series of dual-feasible solutions, and therefore produces a series of iteratively improving lower bounds. Each dual-feasible solution is obtained by optimizing over an inner approximation of the dual polytope. This approximation is obtained by linking groups of dual variables by linear constraints, leading to a problem of smaller dimension. The inner approximation is iteratively refined by splitting the groups into smaller groups until an optimal dual solution is found.

Keywords: column generation, aggregation, cutting and packing, convergent lower bounds

1 Introduction

This paper is about the cutting-stock problem and the famous model of Gilmore and Gomory [5]. The column generation algorithm is a popular approach for solving this model, as it is used for a large class of problems with a prohibitively large number of variables.

A classical column generation approach typically applies the the following algorithmic template: (i) reach the best dual solution inside the current dual polytope (defined by a limited set of constraints), (ii) find a valid dual constraint (primal column) that is violated by the current solution (report optimum if there is no such violated constraint), and (iii) update the current polytope description by adding a violated constraint (therefore refine the current description) and repeat from (i). As such, the column generation method constructs at each step an *outer* approximation of the dual polytope and optimizes over this polytope. Column generation algorithms are known to have convergence issues: in many

cases, one needs many steps before obtaining a valid lower bound for the problem. Several works have been dedicated to the stabilization of the column generation (reduction of the number of steps). See for example [4, 6, 8, 1, 3].

For the cutting-stock problem, a now classical approach of computing fast lower bounds is based on so-called *dual-feasible functions* (DFF) ([7], see also [2] for a survey of this subject). Those functions are related to valid dual solutions of the . This approach provides excellent results for cutting-stock problems which a regular structure, but is difficult to generalize, even for variants of the cutting-stock. Furthermore, to our knowledge if the set of functions used does not contain a function that yields the optimal dual solution, no convergent methods based on DFF exist. Therefore, an alternative method for computing fast lower bounds would be useful.

We propose a new method that can provide fast lower bounds for the cutting-stock problem, while converging toward the optimum of the column-generation model. The basic idea is to iteratively construct an *inner* approximation of the dual polytope, whose structure is simpler, that is optimized, and then refined until the optimum of the Gilmore-Gomory model is reached. This inner approximation of the polytope is obtained by adding additional non valid cuts that link dual variables that are gathered into groups. The resulting polytope has less faces and less variables, and so, it is more easily optimized; furthermore, it can be directly lifted to polytope included in the initial dual polytope.

The added cuts enforce the dual coefficient in a group to be on a line. This allows us to reformulate the simplified problem with a new model whose variables are the slope and y-intercept of each group.

The step-by-step algorithmic template is the following: (i) construct the restricted inner dual polytope (initial approximation); (ii) reach the best solution inside the current restricted dual polytope ; (iii) find a non valid constraint in the restricted polytope that is too tight ; (iv) replace this constraint with a new weaker constraint in the approximate polytope (construct a refined restricted polytope) and repeat from (ii). Fig. 1 presents an intuitive illustration of this method.

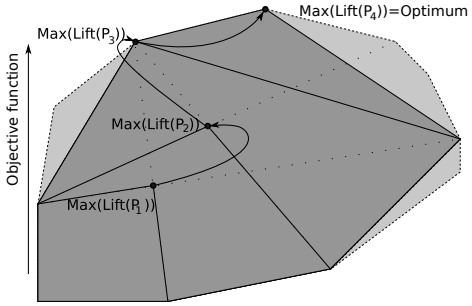

Fig. 1. Our method uses an "inner approximated dual polytope" that is iteratively refined ($\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, ...) so as to reach the optimum of an initial polytope $\mathcal{P}$ with prohibitively many constraints. Each $\mathcal{P}_k$ has fewer variables and fewer constraints then $\mathcal{P}$ and it can be lifted to a polytope completely included in $\mathcal{P}$; the optimum of any intermediate $\mathcal{P}_k$ is a lower bound for the $\mathcal{P}$ optimum.

The remainder is organized as follows: Section 2 describes the inner dual polytope built at each iteration. Section 3 is devoted to the relations between the knapsack polytope and our inner dual polytope, so as to explain how our linearity relations lead to overly-tight constrains. Section 4 deals with the convergent algorithm, and Section 5 contains a summary of our computational experiments, followed by conclusions.

2 The restricted dual polytope model

Let us first recall the well-known Gilmore-Gomory model [5] and introduce notations. Consider n items of weights defined by vector $w = [w_1\ w_2\ \dots w_n]^T \in \mathbb{Z}_+^n$. Given a capacity C, a cutting-pattern is an integer column vector $a = [a_1 \dots a_n]^T \in \mathbb{Z}_+^n$ such that $a^T w \leq C$. Without loss of generality, we consider $w_i \leq C, \forall i \in \overline{1,C}$. The set $\overline{K}$ of all cutting-patterns is defined by the (exponentially many) integer feasible solutions of a knapsack problem with capacity C. The cutting-stock problem requires finding the minimum number of cutting-patterns one can select so as to include each item $i \in \overline{1,n}$ at least b_i ($b_i \in \mathbb{Z}_+$) times. The dual of the linear relaxation of the model can be written using a vector $y = [y_1\ y_2\ \dots\ y_n]^T \in \mathbb{R}_+^n$ of decision variables and a (possibly exponential) number of dual constraints (primal columns).

$$\left.\begin{array}{c} \max b^T y \\ a^T y \leq 1, \forall a \in \overline{K} \\ y_i \geq 0, \quad i \in \overline{1,n} \end{array}\right\} \mathcal{P} \tag{2.1}$$

2.1 Formal description of our dual approximated polytope

We now describe our approach for computing a valid inner approximation of model (2.1). The rationale behind our method is to enforce the dual solution to have a certain shape, i.e., to follow a piecewise (groupwise) linear function. The parameters of the different pieces of this function become decision variables.

We consider a partition of the set of items $I = \{1, 2 \dots n\}$ into k groups indexed by $j \in \overline{1,k}$. Constraints are added to the dual values in such a way that dual values in a given group j satisfy a linearity constraint depending on a slope α^j and a y-intercept β^j; α^j and β^j will become decision variables in our model. More precisely, if item i is in group j, then $y_i = \alpha_j w_i + \beta_j$. We introduce the following notations. Group j contains items I^j ($n_j = |I^j|$); We denote by y^j, w^j, b^j and a^j the vectors related to dual variables, item sizes, demands of items and, respectively, pattern variables in I^j. A dual solution $[y_1 \dots y_n]$ can be rewritten as concatenation of k group components: $[y^1 \dots y^k]$.

Let us re-write (2.1) using only variables α^j and β^j. We first re-write the objective function. For each group j, we consider y^j as a linear function of w^j and $\mathbf{1}_{n_j}$ (vector $[1\ 1 \dots\ 1]^T$ with n_j elements): $y^j = \alpha^j w^j + \beta^j \mathbf{1}_{n_j}$. Observe that

$$b^T y = \sum_{j=1}^{k} \left(b^j\right)^T y^j = \sum_{j=1}^{k} \left(b^j\right)^T \left(\alpha^j w^j + \beta^j \mathbf{1}_{n_j}\right)$$

The constraint aggregation is a key step in reducing the number of constraints. Any $\mathcal{P}$ constraint from (2.1) can be written as $\sum_{j=1}^{k} \left(a^j\right)^T y^j \leq 1$. Let us focus on term j and observe that

$$\left(a^j\right)^T y^j = \left(a^j\right)^T \left(\alpha^j w^j + \beta^j \mathbf{1}_{n_j}\right) = \alpha^j \left(\left(a^j\right)^T w^j\right) + \beta^j \left(\left(a^j\right)^T \mathbf{1}_{n_j}\right) \tag{2.2}$$

The coefficient of α^j is $c^j = (a^j)^T w^j$ and represents the total weight of the I^j items selected by cutting-pattern a. The goal of the proposed model is to aggregate all $\mathcal{P}$ constraints ($a^T y \leq 1$) for which $(a^j)^T w^j$ is equal to a given c^j, for *any* group $j \in \overline{1,k}$. Even when $(a^j)^T w^j$ is fixed, the coefficient of β^j is not fixed in (2.2), *i.e.*, it is $\left(a^j\right)^T \mathbf{1}_{n_j} = \sum_{i \in I^j} a_i^j$. However, the variation of this value is bounded by the maximum and minimum number of items of group j that fit exactly each possible size c: $M(j,c) = \max\left\{\sum_{i \in I^j} x_i : \sum_{i \in I^j} w_i x_i = c,\ x_i \in \mathbb{Z}_+\right\}$ and $m(j,c) = \min\left\{\sum_{i \in I^j} x_i : \sum_{i \in I^j} w_i x_i = c,\ x_i \in \mathbb{Z}_+\right\}$. If there is no feasible solution for the subset-sum problem associated with size c and set I^j, we say that the values $M(j,c)$ and $m(j,c)$ are *undefined*.

We are now ready to express model (2.1) with variables α and β only.

$$\left.\begin{array}{r} \max \sum_{j=1}^{k} \left(b^j\right)^T w^j \alpha^j + \left(b^j\right)^T \mathbf{1}_{n_j} \beta^j \\ \sum_{j=1}^{k} c^j \alpha^j + \mathcal{M}(j,c^j)\beta^j \leq 1,\ \forall c^1 \ldots c^k \in \mathbb{Z}_+ \text{ s.t. } \sum_{j=1}^{k} c^j \leq C \\ \text{with } \mathcal{M} \in \{M, m\} \text{ defined on } c_j \\ \alpha^j \in \mathbb{R} \\ \beta^j \in \mathbb{R} \end{array}\right\} \mathcal{P}_k \tag{2.3}$$

This model has $2k$ real variables (α^j and β^j with $j \in \overline{1,k}$) that can be either *positive* or *negative*. Since it is well-known that the dual solution of the standard cutting-stock problem is non-decreasing (see, *e.g.,* [1]), a non-compulsory constraint $\alpha^j \geq 0$ can be added. The main $\mathcal{P}_k$ constraints (hereafter referred to as *capacity constraints*) are generated from values $c^1, c^2, \ldots c^k$ with $\sum_{j=1}^{k} c^j \leq C$ such that M and m defined on all $c^j, \forall j \in \overline{1,k}$. Such a constraint is only useful if it uses $\mathcal{M} = M$ when $\beta^j \geq 0$ (positive β^j side) and $\mathcal{M} = m$ when $\beta^j \leq 0$.

Let us first formally show that any solution of (2.3) corresponds with a valid solution for (2.1).

Proposition 1. *Any feasible $\mathcal{P}_k$ solution in above model* (2.3) *can be lifted on a feasible solution of $\mathcal{P}$ in model* (2.1)*. The lifted solution $y \in \mathcal{P}$ is directly obtained by composing $[y_1\ y_2\ \ldots y_n]$ as $[y^1\ y^2 \ldots y^k]$, with $y^j = \alpha^j w_i^j + \beta^j$, $\forall j \in \overline{1,k}$.*

Proof. Assume that the lifted solution y does not belong to P. There needs to exist a valid pattern a such that $\sum_{j=1}^{k} \sum_{i=1}^{n_j} a_i^j y_i^j > 1$. By replacing y_i^j with $\alpha^j w_i^j + \beta^j$, we obtain $\sum_{j=1}^{k} \sum_{i=1}^{n_j} a_i^j (\alpha^j w_i^j + \beta^j) > 1$. For each j, let $c^j = \sum_{i=1}^{n_j} a_i^j w_i^j = \left(a^j\right)^T w^j$. The values $M(j,c^j)$ and $m(j,c^j)$ are defined, and $m(j,c^j) \leq \sum_{i=1}^{n_j} a_i^j \leq M(j,c^j)$; as such, $\sum_{i=1}^{n_j} a_i^j (\alpha^j w_i^j + \beta^j) \leq c^j \alpha^j + \mathcal{M}(j,c^j)\beta^j$, where $\mathcal{M} = M$ if $\beta^j \geq 0$ and $\mathcal{M} = m$ if $\beta^j < 0$. If follows that $\sum_{j=1}^{k} c^j \alpha^j + \mathcal{M}(j,c^j)\beta^j > 1$, and so, the $\mathcal{P}_k$ solution $[\alpha, \beta]$ is not valid. Therefore, if a $\mathcal{P}_k$ solution is lifted to a non-feasible $\mathcal{P}$ solution, then the initial $\mathcal{P}_k$ solution can not be feasible in $\mathcal{P}_k$.

The size of model (2.3) only depends on C and k, and not on n. Generally speaking, the number of constraints may be exponentially large (in k). However, if one only wants to generate fast lower bounds, one can consider k as a constant: in this case, the number of constraints is limited by the pseudo-polynomial bound $O(C^k)$—not always reached in practice due to dominance reasons presented next.

2.2 Computing m and M values

A considerable part of the capacity constraints are redundant. Consider any constraint generated by $c^1 \dots c^k$ and focus on a group j. Any constraint generated by replacing c^j with a $c'^j < c^j$ such that $M(j, c'^j) \leq M(j, c^j)$ (a reasonable and frequent situation), decreases term j in the capacity constraints of $\mathcal{P}_k$ in (2.3). Indeed, since $\alpha^j \geq 0$ and the M terms only arise in constraints using positive β^j, it follows that $\alpha^j c'^j + \beta^j M(j, c'^j) \leq \alpha^j c^j + \beta^j M(j, c^j)$. As such, the new constraint replacing $(c_j,\ M(j, c'^j))$ with $(c'^j,\ M(j, c^j))$ is redundant.

More generally, we show that, given a set of pairs (2-dimensional vectors) $(c'^j, \mathcal{M}(j, c'^j))$ (with $\mathcal{M} \in \{M, m\}$ and $c'^j \leq c^j$), non-dominated constraints can be associated *only to vertices of the convex hull* of this set of pairs. The proof results from the fact that any pair $(c'^j, \mathcal{M}(j, c'^j))$ that is not a vertex can be written as a convex combination of other vertex pairs, i.e. $(c'^j, \mathcal{M}(j, c'^j)) = \sum_{i=1}^{\ell} \lambda_i \cdot (c_i^j, \mathcal{M}(j, c_i^j))$ where $\lambda_i \geq 0, \forall i \in \overline{1,\ell}$ and $\sum_{i=1}^{\ell} \lambda_i = 1$. The same multipliers λ_i can be applied to express any capacity constraint in which the term j is $\alpha^j c'^j + \beta^j M(j, c'^j)$ as a convex combination of ℓ capacity constraints in which the term j is associated to the above vertex pairs. The $\lambda-$weighted sum of these ℓ constrains results in the initial constraint: (A) term j in the left-hand side is $\alpha^j c'^j + \beta^j M(j, c'^j)$ (resulting from the convex expression of $\alpha^j c'^j + \beta^j M(j, c'^j)$ and (B) the right-hand side is 1 (using $\sum_{i=1}^{\ell} \lambda_i = 1$).

Finally, note that for a given set I_j, the dynamic program only has to be run twice to compute all possible values $M(j, c)$ and $m(j, c)$, $c = 1, \dots, C$. Such dynamic programs process one by one the items of I_j, and so, a part of the numerical results (corresponding to a subset of I_j) can be re-used when I_j is split (in Section 4).

2.3 Optimization of the model (2.3)

2.3.1 Classical optimization via full $\mathcal{P}_k$ construction for small k

For a small value of k, the model is rather small. The main optimization questions regard the constraints. Given $c_1, \dots c_k$, there is no need to generate capacity constraints using all (2^k) combinations of $M(j, c^j)$ and $m(j, c^j)$ ($\forall j \in \overline{1,k}$), because term j can be written using $\mathcal{M} = M$ when $\beta^j \geq 0$, or $\mathcal{M} = m$ when $\beta^j \leq 0$. To generate only the necessary M/m combinations, the optimization method proceeds as follows. Given a initial solution $\alpha^1, \beta^1, \dots, \alpha^k, \beta^k$ (0 in the very first step), generate only the capacity constraints with the M/m functions corresponding to the current β^j values and optimize. If the resulting solution changes certain β^j sides, update the corresponding constraints as necessary (e.g., replace M by m for any initially positive β^j that becomes strictly negative) and re-optimize. The process is repeated while M/m changes are necessary.

2.3.2 Optimizing $\mathcal{P}_k$ for larger values of k

The number of constraints grows exponentially with the value of k. For larger values of k, one needs to generate these constraints when needed, using a column generation scheme; the (2.3) constraints are constructed by solving a pricing problem. A main difference with the original column generation model is that

the pricing problem is applied on the groups instead of on each item. One can reuse the information obtained when values $M(j,c)$ and $m(j,c)$ are computed.

The objective of the pricing problem is to determine the values $c^1,\ldots,c^k$ that maximise $\sum_{j=1}^{k} \alpha^j c^j + \beta^j \mathcal{M}(j,c^j)$ under the constraint $\sum_{j=1}^{k} c^j \leq C$. This problem can be formulated as a multiple-choice knapsack problem: for each group j, and for each value c^j where $\mathcal{M}(j,c^j)$ is defined, create an item of size c^j and value $\mathcal{M}(j,c^j)$. Then a dummy item of size $c_j = 0$ and value 0 is added for each group j. The objective is to select one item per group in such a way that the total size does not exceed C. This problem can be solved by dynamic programming.

This scheme can be used to (more) rapidly determine the optimum of (2.3) for fixed choices of M/m at each group. As in Sec. 2.3.1 above, the current optimizer can be called several times, once for each necessary M/m change.

3 Constraints of the restricted dual polytope and the knapsack polytope

This section explores certain theoretical properties of the dual approximated polytope $\mathcal{P}_k$, *e.g.*, we investigate the impact of imposing linearity constraints over the items of a group with respect to imposing equalities (as in classical dual cuts). For this, we study the relations between $\mathcal{P}_k$ and the classical knapsack polytope. This provides a better vision for certain ideas applied in the iterative convergent method from Sec. 4. Let us precisely define the knapsack polytope $\mathcal{K}$ as the convex hull of the set of integer solutions $\overline{K}$ of the knapsack problem:

$$\begin{array}{c} \mathcal{K} = \text{Conv}\{\overline{K}\}, \text{ where} \\ \overline{K} = \{a \in \mathbb{Z}_+^n : w^T a \leq C\} \end{array} \tag{3.1}$$

Let us focus on the artificial dual cuts (or, equivalently, artificial cutting-patterns) implicitly introduced in (2.1) by our linearity constraints. These linearity constraints implicitly impose new dual cuts, which can also be seen as *exchange vectors* (see [10]).

In the existing literature, exchange vector are of the form $y_i \leq y_j$. Such (in)equalities can be seen as exchange vectors in the sense that they can be combined with existing cuts to generate new artificial cuts, *e. g.* if $y_i \leq y_j$, any cut involving y_j implicitly generates a new cut by replacing y_j by y_i. In this case, the direct replacements lead to artificial cuts with integer coefficients that would be easy to detect. We show that our linearity-enforcing constrains lead to non-valid artificial cuts that do *not* have integer coefficients. Furthermore, the patterns associated to our cuts always belong to the fractional knapsack polytope, which is not the case for simple $y_i = y_j$ constraints. This means that the degree of infeasibility of the artificial cuts generated from linearity constraints is smaller than that generated by basic equalities between variables.

We now formally show these results in the following three propositions.

Proposition 2. *A solution $a \in R_+^n$ belongs to $\mathcal{K}$ if and only if $a^T y \leq 1$ is a valid constraint of $\mathcal{P}$.*

Proof. The statement $a \in \mathcal{K} \Rightarrow a^T y \leq 1, \forall y \in \mathcal{P}$ *directly follows from the definition* (2.1) *of* $\mathcal{P}$*. While the inequalities of* $\mathcal{P}$ *are defined only using integer solutions of* $\mathcal{P}$*, any* $a \in \mathcal{K}$ *can be obtained as a convex combination of certain integer solutions. This convex combination can also be applied on the inequalities associated to the integer solutions, leading to a inequality that corresponds to* a*.*

Next, it remains to prove that $a^T y \leq 1, \forall y \in \mathcal{P} \Rightarrow a \in K$*. For this, we will show that for any solution* $\hat{a} \in \mathbb{R}^n_+$ *such that* $\hat{a} \notin \mathcal{K}$*, one can construct* $\overline{y} \in \mathcal{P}$ *such that* $\hat{a}^T \overline{y} > 1$*. For this, we will first generate* $\overline{y} \in \mathbb{R}^n$ *such that* $\hat{a}^T \overline{y} > 1$ *and then we will show that* $\overline{y}$ *is a valid solution of* $\mathcal{P}$*.*

To generate $\overline{y}$*, let us first construct a solution* $\overline{a}$ *belonging to a* $\mathcal{K}$ *facet that separates* $\hat{a}$ *from* $\mathcal{K}$*. This solution* $\overline{a}$ *is obtained by simply "scaling"* $\hat{a}$ *down, i. e.,* $\overline{a} = s\hat{a}$*, where* s *is the maximum scale factor* $s < 1$ *such that* $s\hat{a} \in \mathcal{K}$*. This maximum scale factor needs to be strictly positive, as* $\hat{a} \geq 0_n$ *and* $\mathcal{K}$ *contains the polytope delimited by* $\sum_{i=1}^{n} a_i \leq 1$ *(recall that we only use items smaller than* C*). Let us focus on the relation between* $\overline{a}$ *and* $\mathcal{K}$*: since* s *is maximum, there exists a facet that contains* $\overline{a}$ *and that cuts* $\hat{a}$ *from* $\mathcal{K}$*. The facet equation could not have the form* $\overline{y}^T a = 0$*, because such a facet would contain both* $\overline{a}$ *and* $\hat{a}$*. As such, the facet equation can be written as* $\overline{y}^T a = 1$ *for a certain* $\overline{y} \in \mathbb{R}^n$ *not necessarily positive;* $\overline{a}$ *belongs to this facet, i.e.,* $\overline{a}^T \overline{y} = 1$*. Since* $\hat{a} = \frac{\overline{a}}{s}$ *and* $s < 1$*, we obtain* $\hat{a}^T \overline{y} = \overline{y}^T \hat{a} = \frac{\overline{y}^T \overline{a}}{s} > \overline{y}^T \overline{a} = 1$*.*

It remains to show that $\overline{y}$ *is a valid* $\mathcal{P}$ *solution: we need to show that: (a)* $\overline{y}$ *verifies the* (2.1) *knapsack inequalities of the form* $a^T y \leq 1, \forall a \in \overline{K}$*, and (b)* $\overline{y}$ *is non-negative. Regarding the knapsack inequalities, observe that, given any* $a \in \mathcal{K}$*, the inequality* $a^T \overline{y} \leq 1$ *is valid, because* $\overline{y}$ *determines the* $\mathcal{K}$ *facet* $\overline{y}^T a \leq 1, \forall a \in \mathcal{K}$ *(verified with equality by the solutions of the facet). Finally, we only need to check that* $\overline{y} \in \mathbb{R}^n_+$*. Assume there is* $\overline{y}_i < 0$ *and consider a solution* a' *of the facet (i.e.,* $\overline{y}^T a' = 1$*) such that* $a'_i > 0$*. If no such* a' *existed, the facet equation could be written in the form* $a'_i = 0$ *(recall* $\mathcal{K}$ *is fully dimensional), a case already excluded. By decreasing* a'_i *to* 0*, one could obtain a valid* $\mathcal{K}$ *solution that would be separated by the facet, which is impossible. This shows that all components of* $\overline{y}$ *are non-negative. Since* $\overline{y}$ *also verifies all* $\mathcal{P}$ *inequalities,* $\overline{y}$ *needs to belong to* $\mathcal{P}$*. We constructed for a given* $\hat{a} \notin \mathcal{K}$ *a solution* $\overline{y} \in \mathcal{P}$ *such that* $\hat{a}^T \overline{y} > 1$*.*

Proposition 3. *Given any* $y \in R^n_+$*, then* $y \in \mathcal{P}$ *if and only if* $y^T a \leq 1$ *is a valid cut of* $\mathcal{K}$*.*

Proof. The fact that all $y \in \mathcal{P}$ *yield valid inequalities for* $\mathcal{K}$ *has already been discussed in the literature [3]. Conversely, as already described above, if* $y^T a \leq 1$*,* $\forall a \in \overline{K}$*, then, since* $y \in \mathbb{R}^n_+$*, all constraints of* (2.1) *are satisfied by definition.*

The above propositions lead us to the following conclusion.

Proposition 4. *Any non-valid* $\mathcal{P}$ *cut* $\hat{a}^T y \leq 1$ *(*$\hat{a} \geq 0_n$*) that cuts off a solution* $\overline{y} \in \mathcal{P}$ *(i.e.,* $\hat{a}^T \overline{y} \not\leq 1$*) is an infeasible knapsack solution* $\hat{a} \notin K$ *separated from* $\mathcal{K}$ *by a valid knapsack cut* $\overline{y}^T a \leq 1$ *and vice-versa.*

Proof. First, consider a non-valid $\mathcal{P}$ cut $\hat{a}^T y \leq 1$. Since $\hat{a}^T \overline{y} > 1$ for $\overline{y} \in \mathcal{P}$, Proposition 3 states that $\overline{y}$ generates the valid $\mathcal{K}$ cut $\overline{y}^T a \leq 1, \forall a \in \mathcal{K}$; this cut separates $\hat{a}$ from K. Conversely, if $\hat{a} \notin \mathcal{K}$, using Proposition 2 we deduce that $\hat{a}^T y \leq 1$ cannot be a valid constraint for $\mathcal{P}$, i.e., $\exists \overline{y} \in \mathcal{P}$ such that $\hat{a}^T \overline{y} > 1$, yielding the non-valid $\mathcal{P}$ cut $\hat{a}^T y \leq 1$ that separates $\overline{y} \in \mathcal{P}$.

The above proposition can be useful for evaluating the "infeasibility degree" of an artificial non-valid dual cut. When introducing non-valid cuts, one might be interested in the size of the $\mathcal{P}$ zone cut off by the non-valid cut, or, more importantly, on the number of $\mathcal{P}$ vertices separated by this non-valid cut. Using Proposition 4, we observe a bijection between the $\mathcal{P}$ solutions $\overline{y}$ that are cut off by the non-valid cut $\hat{a}^T y \leq 1$ and the valid knapsack cuts $\overline{y}^T a \leq 1$ separating $\hat{a} \notin \mathcal{K}$. To evaluate the $\mathcal{P}$ zone cut off by the cut, one might evaluate the valid knapsack cuts that separate $\hat{a}$ from $\mathcal{K}$.

Let us compare the infeasible $\mathcal{K}$ solutions generated by linearity constraints with respect to those obtained via equality constraints. The "exchange vectors" of equality constraints are integer and they can directly generate infeasible $\mathcal{K}$ solutions that are integer. A non-valid integer $\mathcal{K}$ solution cannot satisfy the main knapsack inequality, *i.e.*, it does not belong to the fractional knapsack polytope $\left\{y \in \mathbb{R}_+^n \text{ s.t. } w^T y \leq C\right\}$. On the other hand, our approach does not lead to this situation, as proved by below proposition.

Proposition 5. *Any artificial cut resulting from our model (2.3) corresponds to a solution $a \in Z_+^n$ that belongs to the fractional knapsack polytope defined above.*

Proof. Consider that the solution $y_w = \frac{w}{C}$ is linear, and so, it belongs to $\mathcal{P}_1, \mathcal{P}_2, \ldots P_k$. As such, it satisfies any artificial constraint $a^T y \leq 1$ generated by our model. It follows that $a^T \frac{w}{C} \leq 1$ is valid, and so, $a^T w \leq C$ is satisfied.

4 A Convergent Method Using Inner Approximations

For a fixed value of k and a fixed group decomposition, the model (2.3) finds a feasible solution of the dual polytope $\mathcal{P}$. This solution may not be optimal for model (2.1), since (2.3) introduces non-valid $\mathcal{P}$ cuts as described in Sec 3. To obtain a method that converges toward the optimum of (2.1), it is necessary to refine the approximation by relaxing non valid constraints. This can be done by selecting one of the k groups to be split into several subgroups. In practice, the split operation produces two new (sub-) groups, and so, $\mathcal{P}_k$ is refined to $\mathcal{P}_{k+1}$.

The algorithmic template of the convergent method can be summarized as follows: (i) determine a lower bound $\overline{y}$ using any method from Sec. 2.3; (ii) find upper bound y^* using a routine that takes $\overline{y}$ as input (optional, see Sec. 4.2), (iii) select the group to split and the two new (sub-)groups (Sec. 4.2) and (iv) update $\mathcal{P}_k$ to construct $\mathcal{P}_{k+1}$ (Sec. 4.1) and repeat from (i). The stopping condition is one of the following: (a) $\lceil \overline{y} \rceil = \lceil y^* \rceil$ or (b) each group has 2 items or less.

4.1 Updating $\mathcal{P}_k$ constraints to $\mathcal{P}_{k+1}$ constraints

We consider that group j with n_j items is split into groups j_1 (n_{j_1} items) and j_2 (n_{j_2} elements). Although the $\mathcal{P}_k$ description evolves considerably at $\mathcal{P}_{k+1}$, important information can be re-used to optimize $\mathcal{P}_{k+1}$. We first present this process for the classical optimization method from Sec 2.3.1, followed by the column generation scheme from Sec 2.3.2. An important time gain is realized in the second case, as one avoids solving many multiple-choice knapsack problems.

For the classical optimization method, each $\mathcal{P}_k$ constraint involving the $\mathcal{M}(j, c^j)$ term $c^j\alpha^j + \mathcal{M}(j, c^j)\beta^j$ can arise in $\mathcal{P}_{k+1}$ as a constraint involving a $\mathcal{M}(j_1, c^{j_1})$ term and a $\mathcal{M}(j_2, c^{j_2})$ term. In fact, all these new constraints are derived from the old ones by determining, for each c^j, all c^{j_1} and c^{j_2} such that $c^{j_1} + c^{j_2} = c^j$ and $M(j_1, c^{j_1}) + M(j_2, c^{j_2}) = M(j, c^j)$. In practice, one uses a $O(n_{j_1} n_{j_2})$ routine to construct all sub-terms c^{j_1} and c^{j_2} *for all* c^j values.

Regarding the column generation scheme, the lower bound of $\mathcal{P}_k$ can be expressed in the space of $\mathcal{P}_{k+1}$ by considering $\alpha^{j_1} = \alpha^{j_2} = \alpha^j$ and $\beta^{j_1} = \beta^{j_2} = \beta^j$ and keeping the α and β values unchanged for the rest of groups. Recall that the objective of the pricing problem is to find a constraint that is violated by the current values α and β. We propose a first stage of the $\mathcal{P}_{k+1}$ optimization that does not solve this using the classical multi-choice knapsack problem (as in Sec 2.3.2). Instead, the first stage of our extension retrieves all constraints generated for $\mathcal{P}_k$ so as to "adapt" them to $\mathcal{P}_{k+1}$. For each such $\mathcal{P}_k$ constraint, one tries to generate the tightest $\mathcal{P}_{k+1}$ constraint as follows: keep unchanged any constraint term involving groups $j' \neq j$ and apply the multiple-choice knapsack algorithm only using the two groups j_1 and j_2 and a total capacity of $C - \left(\sum_{j' \in \overline{1,k}, j' \neq j} c^{j'}\alpha^{j'} + \mathcal{M}(j', c^{j'})\beta^{j'}\right)$. In many practical cases, the $\mathcal{P}_{k+1}$ optimization is carried out almost instantly because: (1) there are few constraints "inherited" from $\mathcal{P}_k$ and (2) these constraints lead to an upper bound for the $\mathcal{P}_{k+1}$ optimum that is equal to the $\mathcal{P}_{k+1}$ lower bound (this $\mathcal{P}_{k+1}$ lower bound is simply the $\mathcal{P}_k$ optimum).

4.2 Upper bound and Group Split Decisions

Given a lower bound $\overline{y} \in \mathcal{P}^k \subseteq \mathcal{P}$, our proposed upper bounding approach optimizes a polytope $\mathcal{P}^s$ that is described using the same model (2.1) as $\mathcal{P}$, but only with a subset of the constrains of $\mathcal{P}$. More exactly, $\mathcal{P}^s$ only uses $\overline{y}$-*saturated constrains* of $\mathcal{P}$,*i. e.*, classical knapsack constrains ($a^T y \leq 1$) of $\mathcal{P}$ such that $a^T\overline{y} = 1$. Since $\mathcal{P} \subseteq \mathcal{P}^s$, the optimal solution y^* of $\mathcal{P}^s$ yields an upper bound of opt($\mathcal{P}$). The advantage of the proposed approach compared to other upper bounds is twofold: (a) it indicates exactly whether or not $\overline{y}$ is a $\mathcal{P}$ optimum, (see below) and (b) it can be very useful in guiding the group split selection.

Regarding the first aspect above, we show that the lower bound $\overline{y}$ is a $\mathcal{P}$ optimal solution if and only if $b^T y^* = b^T\overline{y}$. We start by a clear implication: if $\overline{y}$ is not optimal, then $b^T y^* \geq \text{opt}(\mathcal{P}) > b^T\overline{y}$. Next, it is enough to show the converse of this implication: if $b^T y^* > b^T\overline{y}$, then $\overline{y}$ is not optimal. Using the fact that y^* respects all $\overline{y}$ saturated constrains, one can show that the segment

joining $\overline{y}$ and y^* contains other feasible $\mathcal{P}$ solutions besides $\overline{y}$. If only $\overline{y}$ would belong to $\mathcal{P}$, then all other solutions of this segment would be cut off from $\mathcal{P}$ by a $\overline{y}$-saturated constraint, which is impossible because y^* satisfies all these constrains. More formally, $\exists \epsilon > 0$ such that the solution $\overline{y} + \epsilon(y^* - \overline{y})$ belongs to $\mathcal{P}$; the objective function value $b^T\overline{y} + \epsilon\left(b^T y^* - b^T\overline{y}\right)$ is strictly between of $b^T\overline{y}$ and $b^T y^*$. If $b^T\overline{y} > b^T y^*$, then $\overline{y}$ is not optimal as one can improve it by moving towards y^* (at least using a small $\epsilon > 0$).

The optimization of $\mathcal{P}^s$ is carried out using a classical column generation scheme that generates only constraints a that maximize $a^T\overline{y}$ and, subject to this, maximise $a^T y$ with respect to the current dual solution y.

Group Splitting Strategy In order to construct a $(k+1)$-partition from a k-partition, one needs to choose a group j to be split into two subgroups j_1 and j_2. Our proposal is to assign the lightest n_{j_1} items to group j_1 and the remaining n_{j_2} items to group j_2. As such, at each extension from k to $k+1$, the goal is to find the pair $(j, n_{j_1}) \in \left\{\left(j', n_{j'_1}\right) : \; j' \in \overline{1,k}, n_{j'_1} \in \overline{1,n_j}\right\}$ that makes the iterative method converge as fast as possible. The impact of certain early (low k) split decisions over all future iterations is difficult to evaluate. Constructing a proven globally optimal split strategy would increase dramatically the computational burden. Therefore, we developed several heuristic indicators (*impact score*) that rapidly evaluate the impact (lower bound increase potential) of a split choice (j, n_{j_1}) at the next $(k+1)$ iteration.

Given a split decision (j, n_{j_1}), the impact score is based on : (A) criteria related to the number of items in group j and their weight span, and (B) the relation between $\overline{y}$ and the upper bound y^* determined as above. The use of y^* can be very useful: one can exploit the property that if $b^T y^* > b^T\overline{y}$, one can find $\epsilon > 0$ such that $\overline{y}_\epsilon = \overline{y} + \epsilon(y^* - \overline{y})$ belongs to $\mathcal{P}$. Furthermore, this solution $\overline{y}_\epsilon$ cannot belong to $\mathcal{P}_k$, as $\text{opt}(\mathcal{P}_k) = b^T\overline{y} < b^T\overline{y}_\epsilon$; $\overline{y}$ can only be separated from $\mathcal{P}_k$ by the linearity constrains of certain groups. The most interesting case is obtained when $y^* - \overline{y}$ is negative over all items of the first (sub-)group and positive over the second. Such a group split would allow the lower bound to advance from $\overline{y}$ to y^*: the first n_{j_1} items are allowed to decrease, and the last n_{j_2} can be increased.

5 Experimental Results

The goal of this section is to globally assess our Iterative Inner Dual Approximation (2IDA) in terms of: (i) quality of the iterative lower bounds with respect to the value of column generation (CG), and (ii) the computational effort needed to reach the optimum of model $\mathcal{P}$. For comparison purposes, we use a classical (non-stabilized) column generation algorithm using dynamic programming for pricing the columns. We used classical cutting-stock instances of the literature (*i.e.,* `VB 1` and `VB 2` from [9], random instances with $C = 10000$ and respectively $n = 10$ and $n = 20$), and three randomly generated instance sets from [2]: `m01`, `m20`, `235` with $n = 100$ and $C = 100$, where the number in the instance name corresponds with the minimum size of an item.

Table 1 reports for each class of instances a comparison between the computing effort of our method (2IDA) and that of the CG algorithm. More detailed results, the source code and the instances are publicly available on-line at `http://www.lgi2a.univ-artois.fr/~porumbel/cs/`. Note that our method is executed without upper bounding on instances `VDB 1`. Since $n = 10$, the optimality stopping condition is to have only groups with 1 or 2 elements. On all other instances, our method uses upper bounding both for optimality testing and for group split selection (Sec. 4.2).

Inst.	Avg. CPU		2IDA vs CG			LB MKP calls			calls UB KP/CG KP		
set	2IDA(LB+UB)	CG	$\prec$	$\simeq$	$\succ$	min	avg.	max	min	avg.	max
`m35`	252(131+121)	265	556	93	351	0	7	86	18/52	63/134	188/316
`m20`	940(647+293)	479	262	82	656	0	85	612	43/92	114/173	235/334
`m01`	1274(599+675)	675	236	65	699	0	24	703	59/93	159/191	322/361
`VB1`	1191(1191+0)	1881	19	0	1	0	20	100	–/11	–/21	–/31
`VB2`	74684(54693+19991)	21165	5	0	15	0	89	206	19/33	32/60	63/95

Table 1. The computing effort of reaching a solution with 2IDA and CG. Columns "Avg. CPU" report the time (in ms.) for 2IDA (in parentheses the CPU time of the lower and the upper bound) and for CG. Columns "2IDA vs. CG" report the number of times 2IDA is (1) faster than CG (column $\prec$), (2) roughly equally fast (difference $< 5\%$, column $\simeq$), and (3) slower (column $\succ$). Columns "LB MKP Calls" provide the number of multiple-choice knapsack algorithm calls required by the $\mathcal{P}_k$ optimization (Section 2.3). The last three columns provide a comparison between the number of columns generated by our upper bounding method (Section 4.2) and CG.

For instances `m35` and `VB1`, 2IDA is better than CG in more cases than the opposite. This is not the case for the other classes, where our method is less competitive. The important computational burden of the upper bound explains that our method may be slower. Note that the number of pricing subproblems to solve remains smaller with 2IDA, even when the computing time is larger.

An important feature of our algorithm is that even when it is slower to converge toward an optimal solution, it has the advantage to produce valid lower bounds from the beginning of the search. This is summarized in Figure 2, where we report the average gap with CG during the execution of CG.

From Figure 2, one can see that on average, our method returns a bound close to the CG optimum within the computing time of CG. With the exception of instances VB2, when roughly 40% of the time needed by CG is needed to obtain a gap under 10%, 2IDA needs roughly 20% of the CG time to obtain 90% of the CG value on average.

6 Conclusions and perspectives

We describe an iterative method for computing lower bounds for the cutting-stock problem. This method converges toward the optimum of the Gilmore and Gomory model by constructing an iterative inner approximation of the dual polytope, instead of an outer approximation as usually done in classical column generation algorithm. It has the advantage of producing good lower bounds in a fast manner.

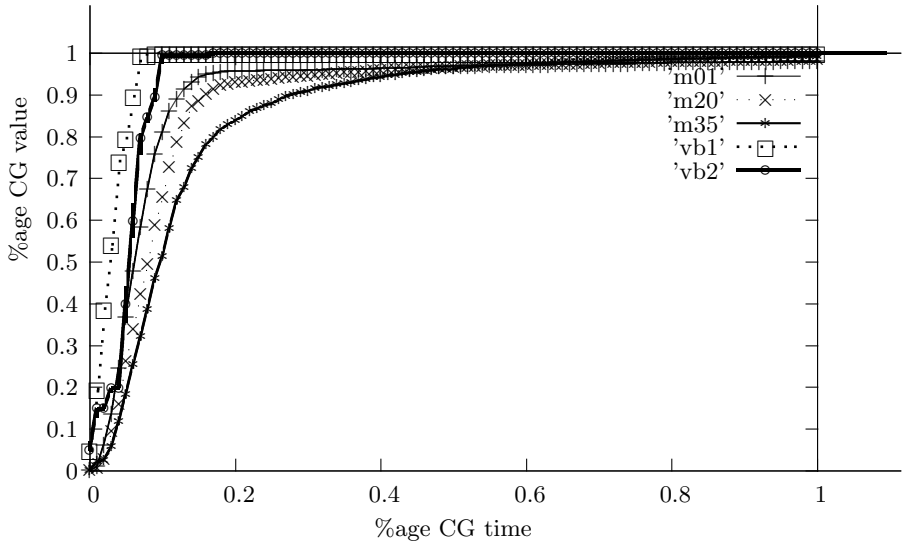

Fig. 2. Evolution of the gap between our lower bound and the value of CG during the (normalized) time needed by CG. These curves are obtained by computing the average gap with CG obtained by 2IDA for $1, 2, \ldots, 100\%$ of the time needed by CG.

The method works by splitting the dual variables into k groups that are aggregated using linear functions. The model obtained is pseudo-polynomial for a given fixed value of k, *i.e.* it does not depend exponentially on the number of items in the data.

It can be noted that certain ideas of the approach can be generalized for other versions of cutting-stock/bin-packing problems, and even for other problems with exponentially many columns—assuming that it is possible to aggregate columns by grouping certain components.

References

1. J. Carvalho. Using extra dual cuts to accelerate column generation. INFORMS Journal on Computing, 17(2):175–182, 2005.
2. F. Clautiaux, C. Alves, and J. Carvalho. A survey of dual-feasible and superadditive functions. Annals of Operations Research, 179(1):317–342, 2009.
3. F. Clautiaux, C. Alves, J. M. V. de Carvalho, and J. Rietz. New stabilization procedures for the cutting stock problem. INFORMS Journal on Computing, 23(4):530–545, 2011.
4. O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. Discrete Mathematics, 194:229–237, 1999.
5. P. Gilmore and R. Gomory. A linear programming approach to the cutting stock problem. 9:849–859, 1961.
6. M. Luebbecke and J. Desrosiers. Selected topics in column generation. Operations Research, 53:1007–1023, 2005.

7. G. S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In 24th Annual Symposium on Foundations of Computer Science, pages 289–297. IEEE, Nov. 1983.
8. R. Marsten, W. Hogan, and J. Blankenship. The BOXSTEP method for large-scale optimization. Operations Research, 23(3):389–405, 1975.
9. F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. Mathematical Programming, 86(3):565–594, 1999.
10. F. Vanderbeck and M. Savelsbergh. A generic view of dantzig-wolfe decomposition in mixed integer programming. Operations Research Letters, 34(3):296–306, 2006.