On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm

François Vanderbeck

Mathématiques Appliquées Bordeaux (MAB), Université Bordeaux 1

351 Cours de la Libération, F-33405 Talence Cedex, France. Fax: +33 (05) 57 96 21 23

Email: fv@math.u-bordeaux.fr Url: http://www.math.u-bordeaux.fr/~fv

September 1995 (Research Paper in Management Studies, 1994-1995, No 29)

(Revised and Submitted to Operations Research in May 1996)

(Revised in May 1997)

(Minor revision in March 1998)

(Accepted for publication in Operations Research)

(Forthcoming in the January-February 2000 issue of Operations Research)

Subject Category

Integer Programming Algorithms: Decomposition, Column Generation, Branch-and-Price.
Production/Scheduling: Cutting Stock / Trim.

Abstract

Dantzig-Wolfe decomposition as applied to an integer program is a specific form of problem reformulation which aims at providing a tighter linear programming relaxation bound. The reformulation gives rise to an integer master problem whose typically large number of variables is dealt with implicitly by using an integer programming column generation procedure, also known as branch-and-price algorithm. There is a large class of integer programs that are well suited for this solution technique. In this paper, we propose to base the Dantzig-Wolfe decomposition of an integer program on the discretization of the integer polyhedron associated with a subsystem of constraints (as opposed to its convexification). This allows to formulate the integrality restriction directly on the master variables and sets a theoretical framework for dealing with specific issues such as branching or the introduction of cutting planes in the master. We discuss specific branching schemes and their effect on the structure of the column generation subproblem. We give theoretical bounds on the complexity of the separation process and the extent of the modifications to the column generation subproblem. Our computational tests on the cutting stock problem and a generalisation, the cutting strip problem, show that, in practice, all fractional solutions can be eliminated using branching rules that preserve the tractability of the subproblem, but there is a tradeoff between branching efficiency and subproblem tractability.

Introduction

The Dantzig-Wolfe decomposition principle of Linear Programming (LP) (Dantzig and Wolfe, 1960) has its equivalent in Integer Programming (IP). Consider an integer program of the form:

$$
[P] \qquad \begin{array}{rcl} \min & & c\,x \\ \text{s.t.} & & \\ A\,x & \geq & b \\ D\,x & \geq & d \\ x & \in & I\!N^n \end{array} \tag{1}
$$

where $A \in \mathbb{Q}^{m \times n}$ and $D \in \mathbb{Q}^{l \times n}$ are rational matrices and $c \in \mathbb{Q}^n$, $b \in \mathbb{Q}^m$, and $d \in \mathbb{Q}^l$ are rational vectors. As we formally show in the next section, one can select a subset of con-

straints that includes the integrality restrictions as a subsystem, $X = \{x \in I\!N^n : Dx \geq d\}$, consider implicitly all the integer solutions in the polyhedron X, and reformulate the original integer program P by replacing its variables with their expression as integer convex combination of a finite set of points and rays of X. Then, one obtains a new integer program, the so-called *master* formulation, that typically has a tighter linear programming relaxation because the reformulation amounts to implicitly taking the convex hull, $conv(X)$, of the integer polyhedron X. Therefore, the master formulation is often better suited for a solution by LP-based branch-and-bound than the original formulation P. However, the master has many more variables and associated columns than the original formulation. Even so, solving the master LP relaxation does not require an explicit enumeration of all its columns since the column generation algorithm allows one to generate columns if and when needed. The column generation subproblem consists in finding a feasible solution to the subsystem that defines a column with minimum LP reduced cost. The algorithm consisting of embedding column generation in a branch-and-bound framework is known as branch-and-price or IP column generation (Barnhart *et al* 1994, Vanderbeck and Wolsey 1996).

To illustrate the IP decomposition principle, let us introduce the following generalization of the cutting stock problem. The model, due to Hurkens (1995), is called the cutting strip problem. It originates from the fabrication of metallic pipes in a make-to-stock manufacturing process. Metal sheets of various dimensions are used as raw material. The pipes are produced by cutting and folding metal strips of appropriate width. Demands for pipes are expressed in length units and are triggered by stock replenishment requirements. Over-production is allowed. The cost of a cutting pattern is proportional to the unused area of cut metal sheets (the leftovers must go back to the furnace). Let there be p different strip widths $w_i \in I\!R_+$ and associated demands $d_i \in I\!R_+$ for $i = 1, \ldots, p$ and K different metal sheets of length L^k and width W^k for $k = 1, \ldots, K$. Let y^k be one if sheet k is used and zero otherwise, and let z_i^k represent the number of strips of width w_i cut from sheet k. Then, the formulation of the cutting strip problem takes the form:

$$[P^{cs}] \qquad \begin{array}{ll} \min & \displaystyle\sum_{k=1}^{K} L^k (W^k\, y^k \; - \; \sum_{i=1}^{p} w_i\, z_i^k) \\ \text{s.t.} & \end{array} \qquad (2)$$

$$\sum_{k=1}^{K} L^k\, z_i^k \;\geq\; d_i \qquad\qquad i = 1, \ldots, p$$

$$\sum_{i=1}^{p} w_i\, z_i^k \;\leq\; W^k\, y^k \qquad\qquad k = 1, \ldots, K \tag{3}$$

$$y^k \;\in\; \{0,1\} \qquad\qquad k = 1, \ldots, K \tag{4}$$

$$z_i^k \;\in\; I\!N \qquad i = 1, \ldots, p, \;\; k = 1, \ldots, K \tag{5}$$

Alternatively, the problem can be formulated in terms of the variables associated with the selection of feasible cutting patterns. Let Q^k be the set of feasible cutting patterns for sheet k, i.e. $Q^k = \{q^k \in I\!N^p : \sum_{i=1}^{p} w_i\, q_i^k \leq W^k\}$, let c_q^k be the cost of pattern $q^k \in Q^k$, i.e. $c_q^k = L^k(W^k - \sum_{i=1}^{p} w_i\, q_i^k)$, and let λ_q^k be the number of times pattern $q^k \in Q^k$ is selected in the solution. Then, the master formulation is an integer program of the form:

$$\begin{aligned}
& & \min \quad & \sum_{k=1}^{K} \sum_{q^k \in Q^k} c_q^k\, \lambda_q^k & & \\
[M^{cs}] & & \text{s.t.} & & & \qquad (6) \\
& \sum_{k=1}^{K} \sum_{q^k \in Q^k} L^k\, q_i^k\, \lambda_q^k \;\geq\; & d_i & & i = 1, \ldots, p \\
& \sum_{q^k \in Q^k} \lambda_q^k \;\leq\; & 1 & & k = 1, \ldots, K \\
& \lambda_q^k \;\in\; & \{0,1\} & \qquad q^k \in Q^k, \;\; & k = 1, \ldots, K
\end{aligned}$$

The master formulation M^{cs} arises from applying the Dantzig-Wolfe decomposition principle to formulation P^{cs} where constraints (3 - 5) are selected as a subsystem, i.e.,

$$X^{cs} = \{(y^k, z^k)_{k=1,\ldots,K} \in I\!N^{K(1+p)} : \sum_{i=1}^{p} w_i\, z_i^k \leq W^k\, y^k \;\text{ for } k = 1, \ldots, K\}$$

$$= \{(y^k, z^k)_{k=1,\ldots,K} \in I\!R^{K(1+p)} : \; y^k = \sum_{q^k \in Q^k} \lambda_q^k, \; z^k = \sum_{q^k \in Q^k} q^k\, \lambda_q^k, \text{ and } \sum_{q^k \in Q^k} \lambda_q^k \leq 1 \;\; \forall k\,,$$

$$\text{and } \lambda_q^k \in \{0,1\} \;\; \forall q^k \in Q^k, \; k\}.$$

The second formulation of the integer polyhedron X^{cs} is based on the enumeration $\{q^k\}_{q^k \in Q^k}$ of all solutions in the integer polyhedron $\{z \in I\!N^p : \sum_{i=1}^{p} w_i\, z_i \leq W^k\}$ for each $k = 1, \ldots, K$ and gives rise to the master reformulation. Solving the LP relaxation

of the master using a column generation procedure involves one column generation subproblem for each metal sheet type $k = 1, \ldots, K$, which takes the form of a knapsack problem:

$$\begin{aligned} \zeta^k(\pi, \mu) = \min \quad & L^k(W^k - \sum_{i=1}^{p}(w_i + \pi_i)z_i) + \mu_k \\ \text{s.t.} & \\ \sum_{i=1}^{p} w_i\, z_i \quad & \leq \quad W^k \\ z_i \quad & \in \quad \mathbb{N} \qquad\qquad i = 1, \ldots, p \end{aligned}$$

where $(\pi, \mu) \in \mathbb{R}_+^{p+K}$ are the dual variables of the master LP relaxation that are associated with the demand covering constraints and the convexity constraints respectively.

The standard cutting stock problem is a special case where all raw sheets are identical and the data can therefore be normalized, i.e., $L^k = L$ for all k, $W^k = W$ for all k, and the demands d_i are expressed in number of standard length L required. Moreover, over-production is typically not rewarded and therefore the cost consists of the number of sheets that are used, i.e. $c_q = 1$ for each non-zero cutting pattern q.

The motivation for using a Dantzig-Wolfe decomposition approach in integer programming is to obtain an improved LP bound by exploiting the structure of the problem: the fact that the integer program over the subsystem X is tractably solvable (although not trivial) is exploited to obtain the bound $LB = \min\{c\,x : Ax \geq b, x \in conv(X)\}$. Lagrangian relaxation, variable redefinition, and cutting planes are alternative approaches that can be used in the same context to yield the same bound, LB. It is well-known that the master LP is the dual formulation of the Lagrangian dual that results from the dualization of the complicating constraints $Ax \geq b$ (Geoffrion, 1974). The variable redefinition approach introduced by Martin (1987) consists in developing an alternative formulation Z for the special structure polyhedron X. In the best of cases, the LP relaxation of Z provides a reformulation of $conv(X)$. Hence, Dantzig-Wolfe decomposition is a special case of the variable redefinition approach where the new variables use for the reformulation Z are the convex combination coefficients. For the cutting strip problem above $Z^{cs} = \{\lambda \in \{0,1\}^{\sum_k |Q^k|} : \sum_{q^k \in Q^k} \lambda_q^k \leq 1\ \forall k\}$. As the number of new variables is exponential in terms of the original problem size, the reformulation is only carried out

implicitly by way of using a column generation procedure. A cutting plane approach is another implicitly way to obtain $conv(X)$. It assumes that the (numerous) inequalities that define $conv(X)$ are known and can be separated upon. When only a partial polyhedral description of $conv(X)$ is known, a column generation approach where the column generation subproblems are solved using a branch-and-cut procedure gives the desired LP bound LB. When the separation problem is a linear program, Martin (1991) shows that there exists a reformulation of P, using auxiliary variables, that gives the same LP-relaxation bound, LB.

There is a large class of integer programs for which Dantzig-Wolfe decomposition has proved to be an efficient solution approach. Gilmore and Gomory (1961 and 1963) have been at the forefront of the development of this approach, with their work on the cutting stock problem. In the last decade, the column generation approach to integer programming has been successfully applied to several problems that can be formulated as some variant of the set partitioning problem with binary variables. These applications include routing problems (Desrosiers *et al* 1994), crew assignment problems (Anbil *et al*, 1993), the generalized assignment problem (Savelsbergh, 1997), edge clustering problems (Johnson, Mehrotra, and Nemhauser, 1993, and Vanderbeck, 1994), and the bin packing problem (Vance *et al*, 1994, and Vanderbeck, 1996) among others. In the present paper, as well as in the recent studies of Vance (1996) and Valerio de Carvalho (1996), cutting stock problems are solved to optimality using column generation within a branch-and-bound algorithm. Such applications, where the master obtained through Dantzig-Wolfe decomposition is an integer program with non-binary variables, emphasize the difficulties inherent to branching in an IP column generation approach.

It is known for a fact that standard branching schemes are not suitable for integer programming column generation. Fixing or bounding a variable of the master requires modifications to the column generation subproblem that generally destroys the structure that is exploited or even required in solving these subproblems efficiently. Moreover, it leads to an unbalanced branch-and-bound tree (see Barnhart *et al* 1994, for instance). Therefore branching schemes have been specially developed for use in the context of a IP column generation algorithm. When the master formulation contains convexity constraints, enforcing that the master solution is made of exactly one column from each

subproblem, integrality can be enforced by adding branching constraints to the subproblem formulation (Desaulniers et al. 1998, Vanderbeck and Wolsey 1996). If the columns and right-hand-side entries of the master are binary, a branching scheme due to Ryan and Foster can be used (see Vance et al. 1994). For the general case involving non-binary master variables, a branching scheme has been proposed by Vanderbeck and Wolsey (1996). In their paper on the bin packing problem, Vance *et al* (1994) also mention a similar branching scheme when they discuss extending their work to the cutting stock problem.

The content of this paper is threefold. In Section 1, we show how the Dantzig-Wolfe decomposition principle can be applied in integer programming. The presentation is based on a complete discretization of the subsystem X which also uses interior points of the integer polyhedron. With this convention (also adopted in Vanderbeck 1994) the integrality restrictions of the original formulation naturally translate into integrality restrictions on the variables of the master. Instead, when the decomposition is based on a convexification of the integer polyhedron X, which has been the common practice, the master variables are not directly restricted to integer values and the integrality requirements are enforced on the convex combinations that arise in the master reformulation (Martin 1987, Barnhart *et al* 1994, Desrosiers *et al* 1994). When the variables of the integer program P are binary, i.e. $X \subseteq \{0,1\}^n$, there are no integer interior points in X. Hence, discretization only makes a difference, compared to convexification, in case some of the variables are general integer, i.e. $X \not\subseteq \{0,1\}^n$. Because the discretization approach establishes a clear relation between the variables of the original formulation and those of the master reformulation, it allows for the development of a unifying and complete theoretical framework to deal with all relevant issues that arise in the implementation of a branch-and-price algorithm, such as that of branching or that of adding cutting planes. When the original integer program includes multiple and identical subsystems, as in the case of the cutting stock problem, the discretization approach can exploit that structure whereas convexification cannot. The issue is important since ignoring the special structure leads to notorious difficulties due to the symmetry in the master. Hence, the discretization approach is also of practical relevance as it permits the development of an efficient branch-and-bound procedure.

In Section 2, we present a branching framework for IP column generation that is an extension of that of Vanderbeck and Wolsey (1996). By considering a wider class of

branching rules, we are able to derive theoretical results concerning the complexity of finding a disjunctive branching constraint that cuts off the current fractional master solution, and hence, we can bound the number of additional constraints and variables that must be added to the column generation subproblem as a result of branching. The branching framework considered here can be broadly described as follows: fractional master solutions are ruled out by carrying out a partition $X = \tilde{X} \cup (X \setminus \tilde{X})$ of the solution space X of the column generation subproblem and by enforcing the integrality of the number of columns that are selected by the master solution in subset $\tilde{X}$ of the partition. We discuss specific ways of defining a partition of the solution space defined by the subsystem. We present the modifications to the column generation subproblem that result from enforcing integrality of the partition cardinality, and we give theoretical bounds on the extent of these modifications. The branching framework encompasses various specific branching schemes that have previously been used in IP column generation applications but also includes new branching rules of practical interest that prove to be computationally efficient and to preserve the tractability of the column generation subproblem.

Finally, in Section 3, we test different branching rules on the cutting stock problem and the cutting strip problem. The computations reveal that, in practice, the extent of the subproblem modifications that result from branching is quite reasonable and much lower than what was predicted by the theory. Indeed, we have been able to obtain an integer solution for all our test problems, using only branching rules that require none or little subproblem modifications. We also show, through a limited comparison of computational results, that the branching rules of general applicability that are proposed here, compare favourably with the more specific schemes used by Vance (1996) or Valerio de Carvalho (1996) for the cutting stock problem. When the original problem consists of multiple non-identical subsystems, as is the case for the the cutting strip problem, the solution space typically exhibits some symmetry. Then, although branching could be implemented simply by bounding the variables of the subproblem, our computational tests indicate that it is more efficient to use more global branching rules as those proposed here. In an independent study on the standard cutting stock problem, Vance (1996) reaches similar conclusions by comparing the use of the standard master formulation to that of a disaggregated formulation where each paper roll is associated with its own subset of master variables.

1 Decomposition of an Integer Program

Consider an integer programming problem of the form (1). The subsystem $X = \{x \in \mathbb{N}^n : D\,x \geq d\}$ is an integer polyhedron. Therefore, it can be generated from a finite set of its points and a finite set of integer rays, i.e. any point in X can be expressed as an integer combination of a finite set of integer points $Q \subseteq X$ and a finite set of integer rays $R \subseteq \mathbb{N}^n$.

Proposition 1 (Nemhauser and Wolsey, 1988, p.104)
If $S = \{x \in \mathbb{R}^n_+ : Dx \geq d\} \neq \emptyset$, $X = S \cap \mathbb{N}^n$, where $(D, d) \in \mathbb{Q}^{l\times(n+1)}$ is a rational matrix, then there exist a finite set of integer points, $Q \subseteq X$, and a finite set of integer directions, $R \subseteq \{r \in \mathbb{N}^n : D\,r \geq 0\} \setminus \{0\}$, such that

$$X = \{x \in \mathbb{R}^n_+ : x = \sum_{q\in Q} q\,\lambda_q + \sum_{r\in R} r\,\lambda_r,\ \sum_{q\in Q} \lambda_q = 1,\ \lambda \in \mathbb{N}^{|Q|+|R|}\}. \tag{7}$$

Note that, when X is bounded, $R = \emptyset$ and Q is the set X itself, i.e.,

$$X = \{x \in \mathbb{N}^n : Dx \geq d\} = \{x \in \mathbb{R}^n_+ : x = \sum_{q\in Q} q\,\lambda_q,\ \sum_{q\in Q} \lambda_q = 1,\ \lambda \in \{0,1\}^{|Q|}\} = \{q\}_{q\in Q}\ .$$

If X is unbounded, R is the set of extreme rays of S (since S is a rational polyhedron, we can assume without loss of generality that all $r \in R$ are integer vectors) and $Q = X \cap \{x \in \mathbb{R}^n_+ : x = \sum_{p\in P} p\,\alpha_p + \sum_{r\in R} r\,\beta_r,\ \sum_{p\in P}\alpha_p = 1,\ \alpha_p \geq 0 \text{ for } p \in P,\ 0 \leq \beta_r < 1 \text{ for } r \in R\}$, where P is the set of extreme points of S. If, moreover, S is a cone ($d = 0$), the points in Q can be seen as rays and included in R, yielding $Q = \emptyset$.

The integer discretization of the subsystem X, i.e. replacing $x \in X$ by $x = \sum_{q\in Q} q\lambda_q + \sum_{r\in R} r\,\lambda_r$, with $\sum_{q\in Q}\lambda_q = 1$ and $\lambda \in \mathbb{N}^{|Q|+|R|}$, yields a new IP formulation known as the master:

$$\begin{array}{lll} & \min & \displaystyle\sum_{q\in Q} c_q\,\lambda_q + \sum_{r\in R} c_r\,\lambda_r \\ [M] & \text{s.t.} & \\ & \displaystyle\sum_{q\in Q} a_q\,\lambda_q + \sum_{r\in R} a_r\,\lambda_r & \geq\ b \end{array} \tag{8}$$

$$\begin{aligned} \sum_{q \in Q} \lambda_q &= 1 \\ \lambda_q &\in \{0,1\} && q \in Q \\ \lambda_r &\in I\!N && r \in R \end{aligned}$$

where $c_q = c\,q \in \mathbb{Q}$ and $a_q = A\,q \in \mathbb{Q}^m$ for all $q \in Q$, and $c_r = c\,r \in \mathbb{Q}$ and $a_r = A\,r \in \mathbb{Q}^m$ for all $r \in R$.

Independent subsystems

When, as is typical to many applications, the matrix D has a block diagonal structure, the decomposition can be carried even further. Let $D^k \in \mathbb{Q}^{t \times p}$ be the k^{th} block of D, for $k = 1, \ldots, K$. To simplify the notation we assume that each block has the same dimensions, thus $l = K\,t$ and $n = K\,p$. Let $c^k \in \mathbb{Q}^p$, $A^k \in \mathbb{Q}^{m \times p}$, and $d^k \in \mathbb{Q}^t$ be the corresponding blocks in c, A, and d. Then, the set X decomposes into K subsystems:

$$X^k = \{x \in I\!N^p : D^k x \geq d^k\}, \quad k = 1, \ldots, K\ ,$$

and the constraints $A\,x \geq b$ can be viewed as linking constraints. Let Q^k and R^k be the finite sets of integer points and directions that characterise X^k.

We set $Q(k) = \{q = (q^k, e^k) \in I\!N^p \times \{0,1\}^K : q^k \in Q^k\} \subseteq I\!N^{p+K}$, where e^k is the k^{th} unit vector ($e^k_i = 0\ \forall i \neq k$ and $e^k_k = 1$). The appending of an incidence vector e^k to column vector q^k is purely for notational convenience: it will allow us to identify whether a vector q belongs to $Q(k)$ by checking whether $q_{p+k} \geq 1$ and hence to treat the case of multiple independent subsystems under the general framework developed for the case of a unique subsystem. We then set $\tilde{Q} = \bigcup_{k=1}^K Q(k)$. $R(k)$ and $\tilde{R}$ are defined similarly. Then, the master formulation takes the form:

$$[\tilde{M}] \qquad \begin{aligned} \min \quad & \sum_{q \in \tilde{Q}} c_q\,\lambda_q + \sum_{r \in \tilde{R}} c_r\,\lambda_r \\ \text{s.t.} \quad & \\ \sum_{q \in \tilde{Q}} a_q\,\lambda_q + \sum_{r \in \tilde{R}} a_r\,\lambda_r &\geq b \\ \sum_{q \in Q(k)} \lambda_q &= 1 \qquad k = 1, \ldots, K \end{aligned} \tag{9}$$

$$\begin{array}{lcll} \lambda_q & \in & \{0,1\} & q \in \tilde{Q} \\ \lambda_r & \in & I\!N & r \in \tilde{R} \end{array}$$

where $c_q = c^k\, q^k \in \mathbb{Q}$ and $a_q = A^k\, q^k \in \mathbb{Q}^m$ for all $q = (q^k, e^k) \in Q(k)$, and $c_r = c^k\, r^k \in \mathbb{Q}$ and $a_r = A^k\, r^k \in \mathbb{Q}^m$ for all $r = (r^k, e^k) \in R(k)$. With these notations, the master formulation $\tilde{M}$ resemble that obtained in the case of an unique subsystem. Note that $\tilde{M}$ could have also been written as $\min\{\sum_{k=1}^K \sum_{q^k \in Q^k} c_q^k \lambda_q^k + \sum_{k=1}^K \sum_{r^k \in R^k} c_r^k \lambda_r^k :\ \sum_{k=1}^K \sum_{q^k \in Q^k} a_q^k \lambda_q^k + \sum_{k=1}^K \sum_{r^k \in R^k} a_r^k \lambda_r^k \geq b;\ \sum_{q^k \in Q^k} \lambda_q^k = 1\ \forall k;\ \lambda_q^k \in \{0,1\}\ \forall q^k \in Q^k, k;\ \lambda_r^k \in I\!N\ \forall r^k \in R^k, k\}$ where the notation are straightforward.

Formulation $\tilde{M}$ arises from the application of Proposition 1 to each integer polyhedron $X^k \subseteq I\!N^p$ for $k = 1, \ldots, K$. Alternatively, Proposition 1 can be applied to $X = \bigotimes_{k=1}^K X^k \subseteq I\!N^{K\,p}$, where $\bigotimes$ denotes the Cartesian product. Then, assuming that X is bounded, $Q = \bigotimes_{k=1}^K Q^k \subseteq I\!N^{K\,p}$ and the resulting master formulation takes the form M (8). The equivalence between M and $\tilde{M}$ is made apparent by the following change of variables:

$$\lambda_{\tilde{q}} = \sum_{q \in Q : q \to q^k} \lambda_q\ , \quad \tilde{q} = (q^k, e^k) \in Q(k) \subset \tilde{Q}\ , \tag{10}$$

$$\lambda_q = \prod_{k=1}^K (\lambda_{\tilde{q}} :\ \tilde{q} = (q^k, e^k) \text{ and } q \to q^k)\ , \quad q \in Q\ , \tag{11}$$

where the notation $q \to q^k$ means that $q^k \in I\!N^p$ is the projection of $q \in I\!N^{K\,p}$ in the space X^k. Equation (11) says that column $q = (q^1, q^2, \ldots, q^K) \in \bigotimes_{k=1}^K Q^k$ is in the solution of M if and only if all its appended sub-columns $\tilde{q} = (q^k, e^k)$ are in the solution of $\tilde{M}$ for $k = 1, \ldots, K$. This change of variables emphazises the benefit of the block diagonal structure in reducing the dimension of the master. Indeed $\tilde{M}$ has $\sum_{k=1}^K |Q^k|$ variables, while M has $\prod_{k=1}^K |Q^k|$ variables.

Identical independent subsystems

Now assume that each block of D has identical characteristics, i.e. $c^k = \overline{c}$, $A^k = \overline{A}$, $D^k = \overline{D}$, and $d^k = \overline{d}$, for $k = 1, \ldots, K$. We set $\overline{X} = \{x \in I\!N^p : \overline{D}x \geq \overline{d}\}$. Let $\overline{Q}$ and $\overline{R}$ be the finite sets of integer points and directions characterising $\overline{X}$. Then, the master takes

the form:

$$
\begin{array}{lrcll}
 & \min & \displaystyle\sum_{q\in\overline{Q}} c_q\,\lambda_q + \sum_{r\in\overline{R}} c_r\,\lambda_r & & \\
[\overline{M}] & \text{s.t.} & & & (12)\\
\displaystyle\sum_{q\in\overline{Q}} a_q\,\lambda_q + \sum_{r\in\overline{R}} a_r\,\lambda_r & \geq & b & & \\
\displaystyle\sum_{q\in\overline{Q}} \lambda_q & = & K & & \\
\lambda_q & \in & I\!N & q\in\overline{Q} & \\
\lambda_r & \in & I\!N & r\in\overline{R} &
\end{array}
$$

where $c_q = \overline{c}\,q \in \mathbb{Q}$ and $a_q = \overline{A}\,q \in \mathbb{Q}^m$ for all $q \in \overline{Q}$, and $c_r = \overline{c}\,r \in \mathbb{Q}$ and $a_r = \overline{A}\,r \in \mathbb{Q}^m$ for all $r \in \overline{R}$.

Observe that $\overline{M}$ and $\tilde{M}$ differ by a change of variables, which, assuming that X is bounded, takes the form:

$$\lambda_{\overline{q}} = \sum_{\tilde{q}\in\tilde{Q}:\tilde{q}\to\overline{q}} \lambda_{\tilde{q}}\,, \qquad \overline{q}\in\overline{Q}\,, \tag{13}$$

where $\tilde{q}\to\overline{q}$ stands for $\tilde{q} = (q^k, e^k) \in I\!N^{p+K}$ and $q^k = \overline{q} \in I\!N^p$ for some k. This transformation is not bijective as there is not a unique way to transform a solution to $\overline{M}$ into a solution to $\tilde{M}$. The aggregation that results from this transformation yields an aggregated convexity constraint $\sum_{q\in\overline{Q}} \lambda_q = K$ and non 0-1 integer variables $\lambda_q \in I\!N$ for $q \in \overline{Q}$.

The cutting strip model

The cutting strip model that we introduced above is an example where the original IP decomposes into K independent subsystems. It falls under the general model P (1). Here $x^k = (y^k, z^k) \in \{0,1\} \times I\!N^p$, $A^k = (0, L^k I) \in \mathbb{Q}^{p\times(p+1)}$, $D^k = (W^k, -w_1, \ldots, -w_p) \in \mathbb{Q}^{1\times(p+1)}$, $b = (d_1, \ldots, d_p) \in \mathbb{Q}^p$, $d^k = 0 \in \mathbb{Q}$, $c^k = L^k D^k \in \mathbb{Q}^{p+1}$, $X^k = \{x^k = (y^k, z^k) \in \{0,1\} \times I\!N^p : \sum_{i=1}^p w_i z_i^k \leq W^k y^k\}$, for $k = 1, \ldots, K$, and $n = K(p+1)$, where I is the identity matrix of size p and y^k, z^k, L^k, W^k, d_i, and p are defined as in P^{cs} (2). Each subsystem X^k is bounded and homogeneous. Therefore, $R^k = \emptyset$, Q^k is X^k itself, and Q^k

contains the null vector, i.e.,

$$Q^k = \{(1,z) : z \in I\!N^p, \sum_{i=1}^{p} w_i z_i \leq W^k\} \cup \{(0,0)\} \ ,$$

for $k = 1, \ldots, K$. Alternatively, the null vector can be excluded from Q^k and the convexity constraints $\sum_{q \in Q(k)} \lambda_q = 1$ replaced by $\sum_{q \in Q(k)} \lambda_q \leq 1$ for $k = 1, \ldots, K$. The sets Q^k then take the form of knapsack polytopes

$$Q^k = \{z \in I\!N^p, \sum_{i=1}^{p} w_i z_i \leq W^k\} \quad \text{for } k = 1, \ldots, K$$

and the master formulation $\tilde{M}$ (9) takes the form M^{cs} (6). The standard cutting stock problem is a special case of the cutting strip problem where all subsystems have identical characteristics, yielding a master of the form $\overline{M}$ (12).

Equivalence between original and master formulations

In the rest of this paper we will refer to P (1) as the compact formulation or the original formulation and M refers to one of the master formulations M (8), $\tilde{M}$ (9), or $\overline{M}$ (12), depending on the context. The compact formulation P and the master formulation M are two models of the same problem. They have the same set of feasible integer solutions and, therefore, the same optimal solution. However, the representation of a solution is different in both formulations and a solution x to P might not correspond to an unique solution λ to M or vice versa:

- If $Q \neq \emptyset$ and $R \neq \emptyset$ (i.e. X is neither bounded nor a cone), and there exists $q \in Q$ and $r \in R$ such that $(q + r) \in Q$, then the transformation from x in P to λ in M is not unique for some points $x \in X$.

- If there are K identical subsystems (model $\overline{M}$), the transformation from λ in $\overline{M}$ to x in P is not unique. Indeed, the master variables have been aggregated through a non-bijective change of variables (13) between $\tilde{M}$ and $\overline{M}$.

The original and master formulations differ however in their linear programming relaxations. The bounds provided by the LP relaxations of P and M are respectively

$$\begin{aligned} Z_{LP}(P) &= \min\{c\,x : Ax \geq b, Dx \geq d, x \geq 0\} \ , \text{ and} \\ Z_{LP}(M) &= \min\{c\,x : Ax \geq b, x \in conv(\{x \in I\!N^n : Dx \geq d\})\} \ . \end{aligned}$$

Thus

$$Z_{LP}(P) \leq Z_{LP}(M) \leq Z_{IP} \ ,$$

where Z_{IP} is the integer solution value ($Z_{IP} = Z_{IP}(P) = Z_{IP}(M)$).

When $conv(\{x \in I\!N^n : Dx \geq d\}) = \{x \in I\!R^n_+ : Dx \geq d\}$ (i.e. when the current formulation of the subsystem already has the integrality property), $Z_{LP}(P) = Z_{LP}(M)$. Then a branching scheme that results in modifying the subproblem can destroy its integrality property. When $conv(\{x \in I\!N^n : Ax \geq b, Dx \geq d\}) = \{x \in I\!R^n_+ : Ax \geq b\} \cap conv(\{x \in I\!N^n : Dx \geq d\})$, $Z_{LP}(M) = Z_{IP}$, and branching is not needed. The applications targeted by this study are cases where the original formulation of the subsystem does not have the integrality property and the master LP does not provide an integer solution and, typically,

$$Z_{LP}(P) < Z_{LP}(M) < Z_{IP}.$$

A decomposition based on the convexification of the subsystem

The traditional presentation of the decomposition of an integer program is based on Minkowski's theorem (cfr Nemhauser and Wolsey, 1988). This approach differs from the above presentation as it consists of reformulating $conv(X)$ and not X itself. The points in $conv(X)$ are expressed as a convex combination of extreme points $q \in \hat{Q}$ and rays $r \in \hat{R}$ of the linear polyhedron $conv(X)$, and the coefficients of this decomposition are not restricted to be integer (see Desrosiers *et al* 1994, for instance). Then, there remains to formulate the integrality restrictions in the master integer program. This is done by using the relationship between the convex combination coefficients λ in M and the original variables x in P to enforce the integrality of x, i.e. constraints

$$x = (\sum_{q \in \hat{Q}} q \ \lambda_q + \sum_{r \in \hat{R}} r \ \lambda_r) \in I\!N$$

are added in the master (Martin 1987, Barnhart *et al* 1994, Desrosiers *et al* 1994).

The two alternative approaches to decomposition, discretization and convexification, give rise to the same LP relaxation of the master. Moreover, the resulting IP master formulations are equivalent when the variables of the original formulation are binary: if $X \subseteq \{0,1\}^n$, then $Q = \hat{Q}$, $R = \emptyset$, and $\sum_{q \in \hat{Q}} q \ \lambda_q \in \{0,1\}^n \Leftrightarrow \lambda \in \{0,1\}^{|Q|}$ as proved

in Barnhart *et al* (1994). When $X \subseteq I\!N^n$ and there is a single subsystem or there are multiple non-identical subsystems, the two approaches might just differ by the way in which an integer solution is represented: for an interior point x^* of X, we can have $x^* = \sum_{q \in \hat{Q}} q\, \lambda_q + \sum_{r \in \hat{R}} r\, \lambda_r \in I\!N^n$ although $\lambda \notin I\!N^{|\hat{Q}|+|\hat{R}|}$. However, when $X \subseteq I\!N^n$ is composed of identical subsystems and the master variable have been aggregated (13), it is not clear how to formulate the integrality constraints when using the convexification approach as there is no unique transformation from $\lambda \in \overline{M}$ to x in P and enforcing the integrality of the aggregated solution, i.e. $\sum_{q \in \hat{Q}} q\, \lambda_q + \sum_{r \in \hat{R}} r\, \lambda_r \in I\!N^n$, does not guarantee the integrality of a corresponding solution x in P. Moreover, Vance's study on the cutting stock (1996) shows that working with the disaggregated formulation (i.e. using formulation $\tilde{M}$ instead of $\overline{M}$) is not efficient.

Alternative reformulations of the subsystem

The Dantzig-Wolfe decomposition approach to integer programming consists in reformulating a subsystem X of problem P according to (7). In turn, this reformulation of X leads to a reformulation of P. The useful characteristic of this reformulation of X is that its linear relaxation gives $conv(X)$ (which explains the quality of the bound provided by the master LP relaxation). However, the integrality property of the subsystem reformulation is obtained at the expense of working with an exponential number of variables, which must be dealt with implicitly by way of using a column generation procedure. As we mention in the introduction, the decomposition of Dantzig-Wolfe is just a special form of reformulation. Alternative reformulation of X with a similar property, namely, whose LP relaxation gives $conv(X)$, can be used to produce the same LP bound on P. We emphasize, however, that we consider reformulations of X as opposed to reformulations of $conv(X)$. With the latter, the same LP bound is obtained, but the integrality restrictions on the original variables does not immediately translate into integrality restrictions for the new variables, hence the drawback outlined above.

In particular, any integer polyhedron $X = \{x \in I\!N^n : Dx \geq d\}$ can be represented by an acyclic directed graph with the property that directed walks from the source node s to the sink node t correspond to solutions of X (see Nemhauser and Wolsey, 1988, p. 312). However the number of nodes in the digraph grows exponentially with the number of con-

straints defining X and grows linearly with the size of the RHS coefficient d. Nevertheless, this network representation has the integrality property. This network reformulation is the one that is being implicitly used in applications where the column generation subproblem is solved by dynamic programming. For instance, if X is the knapsack polytope, the network that underlines the standard dynamic programming procedure consists of n layer whose nodes represent capacity consumption levels that can be achieved using a combination of the first i items, for $i = 1, \ldots, n$, and, a solution is an $s - t$ path in this network. Then, the subproblem solutions, and hence the problem P, can be formulated in terms of the arc flow variables that define an arc incidence vector representation of s-t path solutions.

Eppen and Martin (1987) make explicit use of this network structure and reformulate lot-sizing problems involving only a few time periods in terms of arc flow variables in an appropriate acyclic network. Valerio de Carvalho (1996) solves cutting stock problems using a branch-and-price algorithm where the master is formulated directly in terms of the arc flow variables. He uses a underlying network with $W + 1$ nodes where W is the knapsack capacity, and he places the flow balance constraints in the master (since there is a pseudo-polynomial number of flow balance constraints, they are only added when needed). The column generation procedure consists in solving a knapsack problem and hence results in generating multiple columns: one for each arc used in the path representation of the knapsack solution. We suggest that it might be more practical to exploit the underlying network structure implicitly, as has been done by Desrosiers *et al* (1994). When the arc flow reformulation of P is done explicit as advocated by Martin's variable redefinition approach, the size of the reformulation quickly becomes intractable for realistic applications. When it is used as a basis for a column generation approach, as done by Valerio de Carvalho (1996), the master typically contains a larger number of constraints than reformulation M and hence its intermediate LP relaxations are more difficult to solve. We later show how to exploit the underlying network structure while working with formulation M.

2 Integer Programming Column Generation

We now show briefly how to solve the master IP to optimality. For simplicity of notation, we assume in the rest of this paper that the subsystem X is bounded and therefore $R = \emptyset$. The algorithm is an LP-based branch-and-bound procedure. At each node of the branch-and-bound tree, the master LP relaxation is solved by column generation. If the master LP solution is not integral and the LP bound is not higher than the current incumbent IP solution, branching must take place.

Branching refers to the process of partitioning the solution space to eliminate the current fractional solution. When dealing with the master reformulation, it is well known that a "direct" partitioning of the space $\mathbb{N}^{|Q|}$ by fixing (or bounding) individual variables λ_q is not appropriate as it requires significant alterations to the column generation subproblem and it yields an unbalanced branch-and-bound tree (see e.g. Barnhart *et al* 1994 or Vanderbeck 1994). Instead we take a broader view at branching. We consider a partition of the set Q into $\hat{Q}$ and $Q \setminus \hat{Q}$ and we show that any master fractional solution can be eliminated by fixing (or bounding) the number of points q selected from some subset $\hat{Q} \subseteq Q$. Intuitively, the master solution $\lambda = (\lambda_1, \ldots, \lambda_{|Q|})$ defines a set of weights on the points of Q. If λ is fractional, there must exist a subset $\hat{Q} \subseteq Q$ whose total weight, $\alpha = \sum_{q \in \hat{Q}} \lambda_q$, is fractional and branching takes the form

$$\sum_{q \in \hat{Q}} \lambda_q \leq \lfloor \alpha \rfloor \quad \text{or} \quad \sum_{q \in \hat{Q}} \lambda_q \geq \lceil \alpha \rceil. \tag{14}$$

Branching on an individual variable λ_q corresponds to taking $\hat{Q} = \{q\}$ and yields an uneven partition of Q.

At a branch-and-bound node u, the master LP relaxation takes the form:

$$\begin{array}{lrcll} & \min & \displaystyle\sum_{q \in Q} c_q \, \lambda_q & & \\ [M^u_{LP}] & \text{s.t.} & & & \\ & \displaystyle\sum_{q \in Q} a_q \, \lambda_q & \geq & b & \\ & \displaystyle\sum_{q \in Q^j \subseteq Q} \lambda_q & \leq & K^j & \text{for} j \in G^u \end{array} \tag{15}$$

$$\begin{aligned} \sum_{q \in Q^j \subseteq Q} \lambda_q &\geq L^j && \text{for} j \in H^u \\ \lambda_q &\geq 0 && q \in Q \end{aligned}$$

where G^u and H^u are the sets of cardinality branching constraints of the form (14) that define the problem at node u, i.e. constraints $j \in G^u$ (resp. $j \in H^u$) are characterised by a pair $(Q^j\,,\,K^j)$ (resp. $(Q^j\,,\,L^j)$) where Q^j is a subset of Q and K^j (resp. L^j) is an integer. As the convexity constraints in the master take the same form as the branching constraints, we also include them in G^u and / or L^u. For instance, in the case of K independent subsystems, the convexity constraints are $1 \leq \sum_{q \in Q(k)} \lambda_q \leq 1$ for $k = 1, \ldots, K$. Then, at every node u, the pairs $(Q(k), 1)$ belong to both G^u and H^u for $k = 1, \ldots, K$.

A standard column generation procedure is applied to solve formulation M^u_{LP} at each node u. The formulation initially contains a subset of columns $\mathcal{Q} \subseteq Q$ consisting of all the columns generated at previously processed nodes. This restricted LP formulation $M^u_{LP}(\mathcal{Q})$ is solved. Then a column generation subproblem is solved to price out the remaining columns (in $Q \setminus \mathcal{Q}$). Letting $(\pi, \mu, \nu) \in I\!R_+^{m+|G^u|+|H^u|}$ be an optimal dual solution of the restricted master LP, $M^u_{LP}(\mathcal{Q})$, the reduced cost of column q is:

$$\overline{c}_q = c_q - \sum_{i=1}^{m} \pi_i \; a_{i\,q} + \sum_{j \in G^u}^{m} \mu_j \; g_j(q) - \sum_{j \in H^u}^{m} \nu_j \; h_j(q)\;,$$

where $g(q) \in \{0,1\}^{|G^u|}$ and $h(q) \in \{0,1\}^{|H^u|}$ are indicator functions whose components take values $g_j(q) = 1$ if $q \in Q^j$ and zero otherwise for all $j \in G^u$, and $h_j(q) = 1$ if $q \in Q^j$ and zero otherwise for all $j \in H^u$.

The column generation subproblem

$$\min\{\overline{c}_q : \; q \in Q\}$$

takes the form of an IP:

$$\begin{aligned} & \zeta(\pi, \mu, \nu) = \min && c\,q \;-\; \pi\,A\,q \;+\; \mu\,g \;-\; \nu\,h \\ \text{[SP]} \qquad & \text{s.t.} && \\ & D\,q &\geq\; & d \\ & g &=\; & g(q) \end{aligned} \tag{16}$$

$$
\begin{aligned}
h &= h(q) \\
q &\in \mathbb{N}^n \\
g &\in \{0,1\}^{|G^u|} \\
h &\in \{0,1\}^{|H^u|}
\end{aligned}
$$

where relations $g = g(q)$ and $h = h(q)$ are IP formulations that define the values of boolean vectors g and h as functions of the value of q. If $\zeta(\pi, \mu, \nu) < 0$, a new column $q^* \in \text{argmin}_{\text{q}\in\text{Q}}\overline{\text{c}}_{\text{q}}$ has been found that is added to the master LP formulation M^u_{LP}. Otherwise, the master LP is solved and its value provides a lower bound.

There are many practical issues that are important in designing an efficient implementation of an IP column generation algorithm. For instance, at each branch-and-bound node, an initial feasible LP solution is required. One way to ensure feasibility is to include in the formulation a single artificial column whose entries equal the RHS values for all rows corresponding to a "greater than or equal to" constraint and zero otherwise, and whose cost equals a strict upper bound on the optimal IP value. For a discussion of other ways in which to carry out the phase 1 of the simplex algorithm, of various column generation strategies that may be used (generation of multiple columns, heuristic generation, ...), and of early termination of the column generation procedure, the reader is referred to Barnhart *et al* (1994), Desrosiers *et al* (1994), Vanderbeck (1994), or Vanderbeck and Wolsey (1996). Here, we focus on the design of the branching scheme.

In designing a branching scheme for the IP column generation algorithm one must take on board the standard considerations of efficiency. Branching aims at closing the gap between lower and upper bounds on the solution. With regard to improving the lower bound, an efficient branching rule is one that yields a significant increase in the lower bounds of each child node, i.e. one that results in a balanced branch-and-bound tree even when the solution space presents some symmetry. With regard to improving the upper bound, an efficient branching rule is one that forces the emergence of feasible integer solutions. But, in a column generation approach, there is another very important issue with regard to branching: the assessment of the modifications to the structure of the column reduced costs that result from branching and their consequence on the tractability of the modified column generation subproblem.

Preserving the tractability of the column generation subproblem means that the subproblem should not become significantly more difficult to solve after branching. For the general model P considered in this presentation, the only assumption on the subproblem is that it can be formulated as an integer program. Consequently, a minimum requirement is that the subproblem modifications that arise from the branching scheme can be brought to an integer programming formulation, i.e. that the relations $g = g(q)$ and $h = h(q)$ in (16) can be expressed as IPs. When the column generation subproblem is well suited for a special purpose solution method such as dynamic programming, branching rules should be carefully selected in an effort to preserve the special subproblem structure.

In the rest of this section, we discuss different branching schemes, i.e. we present specific forms of partitions of Q that are sufficient to guarantee that an integer solution to the master will be obtained after adding a finite number of branching constraints (14). For each of them, we consider the practical issues raised above.

2.1 Branching based on partitioning Q with a hyperplane

Proposition 2
Given a feasible solution λ of M^u_{LP} that is not integral, there exists a hyperplane $(\gamma, \gamma_0) \in \mathbb{Z}^{n+1}$ such that $\sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q$ is fractional.

Proof:
Let $F = \{q \in Q :\ \lambda_q - \lfloor \lambda_q \rfloor > 0\} \subseteq Q \subseteq \mathbb{N}^n$. Let $\{q^s\}_{s=1,\ldots,t}$ be the set of extreme points of $conv(F)$. Let $F' = F \setminus \{q^t\}$. As $q^t \notin conv(F')$, there exists a hyperplane $(\gamma, \gamma_0) \in \mathbb{Z}^{n+1}$ that separates $q^t \in \mathbb{N}^n$ from the rational polyhedron $conv(F')$, i.e. $\gamma\, q^t \geq \gamma_0$, but $\gamma\, q \leq \gamma_0 - 1$ for all $q \in Q \cap conv(F')$. Then

$$1 > \sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q - \lfloor \sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q \rfloor = \lambda_{q^t} - \lfloor \lambda_{q^t} \rfloor > 0 \,.$$

■

Branching
Assuming that the solution, λ, of the master LP, M^u_{LP}, at node u is fractional and that a

hyperplane $(\gamma, \gamma_0) \in \mathbb{Z}^{n+1}$ has been found such that

$$\sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q = \alpha \notin I\!N \ ,$$

two new master problem M^v and M^{v+1} are created by adding the branching constraints $\sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q \leq \lfloor \alpha \rfloor$ and $\sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q \geq \lceil \alpha \rceil$ to G^u and H^u respectively, i.e. $G^v = G^u \cup \{(\gamma, \gamma_0, \lfloor \alpha \rfloor)\}$, $H^v = H^u$, $G^{v+1} = G^u$, and $H^{v+1} = H^u \cup \{(\gamma, \gamma_0, \lceil \alpha \rceil)\}$, where the triplet $(\gamma, \gamma_0, \kappa) \in \mathbb{Z}^{n+2}$ defines the new branching constraint.

Subproblem Modifications

To each constraint $j \equiv (\gamma^j, \gamma_0^j, K^j) \in G^u$ is associated a dual variable $\mu_j \in I\!R_+$ in M^u_{LP}. μ_j is added to the reduced cost of columns q satisfying $\gamma^j\, q \geq \gamma_0^j$, making them less attractive for the LP. This penalty is implemented in the column generation subproblem by adding $\mu_j\, g_j$ in the objective, where g_j is a binary variable ($g_j = 1$ if $\gamma^j\, q \geq \gamma_0^j$ and zero otherwise). Then, only the lower bound on g_j needs to be enforced in the subproblem formulation. Thus the relation $g_j = g_j(q)$ takes the form:

$$(\gamma^j_{\max} - \gamma_0^j + 1)\, g_j \geq \gamma^j\, q - \gamma_0^j + 1 \tag{17}$$

where $\gamma^j_{\max} = \max_{q \in Q}\ \gamma^j\, q$.

Similarly, if $j \equiv (\gamma^j, \gamma_0^j, L^j) \in H^u$ and $\nu_j \in I\!R_+$ is the associated master dual variable, a binary variable h_j is introduced that takes value 1 when $\gamma^j\, q \geq \gamma_0^j$, and $\nu_j h_j$ is subtracted from the subproblem objective. Then, only the upper bound on h_j needs to be enforced in the subproblem and the relation $h_j = h_j(q)$ takes the form:

$$(\gamma_0^j - \gamma^j_{\min})\, h_j \leq (\gamma^j\, q - \gamma^j_{\min}) \ , \tag{18}$$

where $\gamma^j_{\min} = \min_{q \in Q}\ \gamma^j\, q$.

Finiteness

As it is shown in the proof of Proposition 2, any fractional master solution can be cut off by partitioning the set of fractional columns F. As the number of possible sets F is finite, the number of extreme points of F that could be singled out is finite, and the number of possible integer bounds on $\sum_{q \in Q:\, \gamma\, q \geq \gamma_0} \lambda_q$ is finite, we must obtain an integer solution

after a finite number of successive applications of this branching scheme.

Finding an appropriate hyperplane
The proof of Proposition 2 suggests that a valid hyperplane can be obtained using the vector $\gamma \in \mathbb{Z}^n$ that characterises the support of the facet of $conv(F \setminus \{q^t\})$ that is the closest to q^t. However, it is not clear that one should proceed along these lines and the question of choosing a hyperplane that meets the requirements of a *good* branching scheme (in the sense defined above) remains open. In a sense the result of Proposition 2 is largely theoretical.

In practice, we will consider hyperplanes with special structure. For instance, when $\gamma = e^i$, the i^{th} unit vector, $\gamma\, q \geq \gamma_0 \Leftrightarrow q_i \geq \gamma_0$ and Q is partitioned into $\{q \in Q : q_i \leq \gamma_0 - 1\}$ and $\{q \in Q : q_i \geq \gamma_0\}$. However, when restricting our attention to a subclass of hyperplanes, we cannot guarantee the existence of a hyperplane (within that subclass) that cuts off the current fractional solution. But, there are weaker results using sets of simple hyperplanes, i.e. component bound constraints.

2.2 Branching based on a set of bounds on the components of q

Here we consider a partition $\hat{Q}$, $Q \setminus \hat{Q}$ of Q, where $\hat{Q}$ is defined in terms of lower and upper bounds on some components of $q \in Q$. A component lower bound constraint, $q_i \geq v$ is defined by a triple $\beta \equiv (i, \geq, v)$, where $i \in \{1, \ldots, n\}$ and $v \in \mathbb{N}$. Let $Q(\beta) = \{q \in Q : \; q_i \geq v\}$ be the set of columns $q \in Q$ that satisfy the component bound constraint β. Similarly, the component upper bound constraint $q_i < v$ is characterised by the triple $\beta \equiv (i, <, v)$. We define the complement β^c of a bound constraint β by replacing $\geq$ by $<$ or vice versa. With this notation,

$$Q = Q(\beta) \cup Q(\beta^c) \quad \text{and} \quad Q(\beta) \cap Q(\beta^c) = \emptyset \; .$$

Let B be a set of component lower and upper bound constraints, then

$$Q(B) = \bigcap_{\beta \in B} Q(\beta) \; .$$

We now show that any master fractional solution can be cut off using a branching constraint based on a partition $Q(B)$, $Q \setminus Q(B)$ of Q for some component bound set B

of small cardinality. We shall need the following notations. Let

$$f(B) = \sum_{q \in Q(B)} (\lambda_q - \lfloor \lambda_q \rfloor) \ .$$

In particular, $f = f(\emptyset) = \sum_{q \in Q} (\lambda_q - \lfloor \lambda_q \rfloor)$ represents the fractionality of the current solution λ of M^u_{LP}.

Proposition 3
Given a feasible solution λ of M^u_{LP} that is not integral, there exists a set of component bounds B with $|B| \leq \lfloor \log f \rfloor + 1$ such that $\sum_{q \in Q(B)} \lambda_q$ is fractional.

The proof of Proposition 3 is based on the following Lemma.

Lemma 4
If, for a given component bound set B, $f(B) \geq 1$, there exists a component bound $\beta \notin B$ such that $0 < f(B \cup \{\beta\}) \leq f(B)/2$.

Proof of Lemma 4
If $f(B) \geq 1$, there exist q^1 and $q^2 \in Q(B)$ with $\lambda_{q^1} - \lfloor \lambda_{q^1} \rfloor > 0$ and $\lambda_{q^2} - \lfloor \lambda_{q^2} \rfloor > 0$. As $q^1 \neq q^2$, $\exists r \in \{1, \ldots, n\}$ such that $q^1_r \neq q^2_r$. Without loss of generality, assume $q^1_r < q^2_r$. Let $v = \lceil \frac{q^1_r + q^2_r}{2} \rceil$, $\beta \equiv (r, <, v)$, and $\beta^c \equiv (r, \geq, v)$. Then, as $q^1_r < v \leq q^2_r$, $f(B \cup \{\beta\}) \geq \lambda_{q^1} - \lfloor \lambda_{q^1} \rfloor > 0$ and $f(B \cup \{\beta^c\}) \geq \lambda_{q^2} - \lfloor \lambda_{q^2} \rfloor > 0$. Moreover, $f(B) = f(B \cup \{\beta\}) + f(B \cup \{\beta^c\})$. Thus either $f(B \cup \{\beta\}) \leq \frac{f(B)}{2}$ or $f(B \cup \{\beta^c\}) \leq \frac{f(B)}{2}$. ∎

Proof of Proposition 3
The proposition can be restated as follows: if $0 < f < 2^l$ where $l \in I\!N$, then $\exists\ B$ with $|B| \leq l$ s.t. $\sum_{q \in Q(B)} \lambda_q \notin I\!N$. We prove this statement by induction. The statement is trivially true for $l = 0$. Assuming that it is true for $0 \leq l < t \in I\!N$, we show that it also holds for $l = t$. Consider $1 \leq f < 2^t$ (the case $f < 1$ being trivial). By Lemma 4, there exists a β such that $0 < f(\{\beta\}) \leq \frac{f}{2} < 2^{t-1}$. Then, by the induction hypothesis, there exists a component bound set B with $|B| \leq t - 1$ s.t. $\sum_{q \in Q(B \cup \{\beta\})} \lambda_q \notin I\!N$. ∎

Branching Scheme and Subproblem Modifications
In a branching scheme based on Proposition 3, the branching constraints $j \in G^u$ or H^u are defined by a pair (B, κ) where B is a set of component bounds and κ is an integer. As

the number of possible component bounds, component bound sets and values κ are finite, an integer solution must be obtained after adding a finite number of branching constraints.

For each branching constraint $j \equiv (B^j, K^j) \in G^u$, there is an associated master dual variable $\mu_j \in I\!R_+$ that comes as an extra cost in the reduced cost of columns $q \in Q(B^j)$. Therefore, a term $\mu_j\, g_j$ is added to the subproblem objective, where $g_j = 1$ if $q \in Q(B^j)$ and zero otherwise, and enforcing the appropriate lower bound on g_j in the subproblem formulation, through the relation $g_j = g_j(q)$, takes the form:

$$\begin{aligned} g_j &\geq 1 - \sum_{\beta \in B^j} (1 - \eta^\beta) && && (19)\\ (q_i^{\max} - v + 1)\, \eta^\beta &\geq (q_i - v + 1) && \forall \beta \equiv (i, \geq, v) \in B^j \\ v\, \eta^\beta &\geq v - q_i && \forall \beta \equiv (i, <, v) \in B^j \\ \eta^\beta &\in \{0, 1\} && \forall \beta \in B^j \end{aligned}$$

where $q_i^{\max} = \max_{q \in Q}\ q_i$ and the binary variables η^β indicate whether or not $q \in Q(\beta)$.

Similarly, for each branching constraint $j \equiv (B^j, L^j) \in H^u$, the corresponding dual variable $\nu_j \in I\!R_+$ represents a reward for selecting a column $q \in Q(B^j)$. Thus, the term $\nu_j\, h_j$ is subtracted from the generic column reduced cost in the subproblem objective and the relation $h_j = h_j(q)$ takes the form:

$$\begin{aligned} h_j &\leq \eta^\beta && \forall \beta \in B^j && (20)\\ v\, \eta^\beta &\leq q_i && \forall \beta \equiv (i, \geq, v) \in B^j \\ (q_i^{\max} - v + 1)\, \eta^\beta &\leq (q_i^{\max} - q_i) && \forall \beta \equiv (i, <, v) \in B^j \\ \eta^\beta &\in \{0, 1\} && \forall \beta \in B^j \end{aligned}$$

Observe that the lower and upper bound constraints on η^β are special cases of (17) and (18) where $(\gamma, \gamma_0) = (e^i, v)$ if $\beta \equiv (i, \geq, v)$ and $(-e^i, 1 - v)$ if $\beta \equiv (i, <, v)$ and e^i is the i^{th} unit vector. Moreover, when $|B^j| = 1$, the above IPs take a simpler form and there is no need to introduce the boolean variable η^β. Indeed, constraint (19) (resp. constraint (20)) can be dropped and g_j (resp. h_j) replaces η^β.

Finding a set B that cuts off the current fractional solution

A polynomial algorithm to find a set B that cuts off the current fractional solution λ is

given implicitly by Lemma 4. Successive application of Lemma 4 provides a set $\overline{B}$ whose cardinality $|\overline{B}|$ is guaranteed to be reasonably small, i.e. $|\overline{B}| \leq \lfloor logf \rfloor + 1$. However, there might exist another set $\tilde{B}$ such that $\sum_{q \in Q(\tilde{B})} \lambda_q \notin I\!N$ and $|\tilde{B}| < |\overline{B}|$.

As the extent of the subproblem modifications increases with $|B|$, one is keen to find a minimal cardinality set B on which to branch. To this end, one can always enumerate all possible sets B in increasing order of their cardinality and stop as soon as a set B has been found for which $\sum_{q \in Q(B)} \lambda_q \notin I\!N$. One only needs to consider component bound constraints $(i, <, v)$ or $(i, \geq, v)$ where $v = \lceil \frac{q_i^r + q_i^{r+1}}{2} \rceil$ and q_i^r and q_i^{r+1} are two consecutive values in the ordered list $0 \leq q_i^1 < q_i^2 < \ldots$ of distinct positive values that are assumed by component i of fractional columns $q \in F = \{q \in Q : \lambda_q - \lfloor \lambda_q \rfloor > 0\}$. Hence, there are $O(n|F|)$ component bound to consider. Since Proposition 3 guarantees the existence of a set B on which to branch with cardinality less than or equal to $\delta = \lfloor logf \rfloor + 1$, the enumeration is over $O((n|F|)^\delta)$ sets B. Checking if a set B yields a fractional sum $\sum_{q \in Q(B) \cap F} \lambda_q$ requires $O(\delta\,|F|)$ operations since checking whether $q \in Q(B)$ requires $O(|B|) = O(\delta)$ operations. Therefore, the complexity of this enumerative algorithm is $O(\delta\,|F|\,(n|F|)^\delta)$, which is exponential. However, in our experiments, we seldom need to consider sets B of cardinality greater than one.

Special case: Branching based on component lower bounds
The branching scheme proposed in Vanderbeck and Wolsey (1996) is a special case of the above, where B consists of lower bound constraints $(i, \geq, v)$ only. There, it is shown that it is always possible to find a partition of the form $\hat{Q}$, $Q \setminus \hat{Q}$ with

$$\hat{Q} = \{q \in Q :\ q \geq p\} \subseteq Q \subseteq I\!N^n \quad \text{for some vector } p \in I\!N^n \tag{21}$$

that yields a branching constraint cutting off the current fractional solution, i.e. such that $\sum_{q \in \hat{Q}} \lambda_q \notin I\!N$. However, the number of component bounds required, i.e. the number of strictly positive components in p, may then be larger than $\lfloor logf \rfloor + 1$. Vanderbeck and Wolsey (1996) also show that when the branching constraint is of the form

$$\sum_{q \in \hat{Q}} \lambda_q \leq 0 \quad \text{or} \quad \sum_{q \in \hat{Q}} \lambda_q \geq K$$

where K is a valid upper bound on $\sum_{q \in \hat{Q}} \lambda_q$, it can be enforced directly in the subproblem by adding a constraint in its formulation and there is no need to amend the expression of

column reduced costs. In particular, whenever the master contains convexity constraints, enforcing that the master solution is made of exactly one column from each subproblem, $K = 1$ is a valid upper bound for all subset $\hat{Q} \subseteq Q(k)$ and branching can always be enforced directly in the subproblem. Vanderbeck and Wolsey (1996) also consider the case of binary columns, $Q \subseteq \{0,1\}^n$. Then, the subproblem modifications can be implemented without introducing additional binary variables η^β. If moreover the right-and-side of the master is the unit vector, $b = 1 \in I\!N^m$, the branching scheme based on (21) reproduces that of Ryan and Foster (a presentation of which can be found in Vance *et al* 1994).

Transforming integer columns into 0-1 vectors

Any bounded integer program can be transformed into a binary integer program. To a vector $q \in Q \subseteq I\!N^n$ corresponds a vector $q' \in \{0,1\}^{n'}$, where $n' = \sum_{i=1}^{n} n_i$, with $n_i = \lceil \log(q_i^{\max} + 1) \rceil$ and $q_i^{\max} = \max_{q \in Q} q_i$. The vector q' associated with q is defined by the relation $q_i = \sum_{l=0}^{n_i - 1} 2^l \, q'_{p_i + l}$ for $i = 1, \ldots, n$ where $p_i = 1 + \sum_{l=1}^{i-1} n_l$. As branching takes a simpler form when the columns are binary vectors (Vanderbeck and Wolsey (1996), it is worth considering a scheme where the column generation subproblem is reformulated as a binary integer program and branching is based on subsets $\hat{Q}$ that are defined in terms of the components of q'.

In fact, any fractional solution can be eliminated by using only such branching rules. Indeed, the scheme used to partition the set Q in the proof of Lemma 4 can be replaced by one based on the components of the associated binary vectors: if $q^1 \neq q^2$, $q^1 \leftrightarrow q'^1$, and $q^2 \leftrightarrow q'^2$, then $\exists\, l \in \{1, \ldots, n'\}$ such that $q'^1_l \neq q'^2_l$, where $q \leftrightarrow q'$ denotes that q' is the binary vector associated with q. It follows that:

Corollary 5

Given a feasible solution λ of M^u_{LP} that is not integral, there exists a subset

$$\hat{Q} = \{q \in Q : q \leftrightarrow q',\ q'_l = 0 \text{ for } l \in I,\ \text{and } q'_l = 1 \text{ for } l \in J\}$$

with I and $J \subseteq \{1, \ldots, n'\}$ and $|I| + |J| \leq \lfloor log f \rfloor + 1$, such that $\sum_{q \in \hat{Q}} \lambda_q$ is fractional.

When columns can be associated with paths in a network

In many applications, the column generation subproblem can be naturally formulated as

a shortest (longest) s-t path problem in an acyclic network. This is in particular the case when the column generation subproblem is solved by dynamic programming (Desaulniers *et al* (1998) discussed several applications of that sort). Then, the columns are associated with s-t paths in the underlying acyclic network, and the master solution can be interpreted as a combination of such paths that satisfies the linking constraints. Let us denote by $q' \in \{0,1\}^{n'}$ the arc incidence vector representation of a path associated with a solution q of the subproblem, where n' is the number of arcs in the underlying network. As before, we use the notation $q \leftrightarrow q'$ to denote that the vector q and q' are two representations of the same solution.

Branching on a single component of q' does not destroy the structure of the subproblem as the resulting modifications simply entail amending the cost of the corresponding arc in the underlying network. Moreover, any fractional solution can be eliminated in this way, i.e., using branching constraints of the form (14) where $\hat{Q} = \{q \in Q : q \leftrightarrow q',\ q'_l = 1 \text{ for some } l \in \{1,\ldots,n'\}\ \}$

Proposition 6
If vectors q' represent arc incidence vectors of s-t paths associated with solutions $q \in Q$ in an appropriate acyclic network representation of the column generation subproblem and the master LP solution λ of M^u_{LP} is not integral, then either an integer solution to M^u can be found by application of the flow decomposition theorem, or there exists an arc $l \in \{1,\ldots,n'\}$ of the underlying network such that the flow along that arc, i.e.

$$\sum_{q \in Q:\ q' \leftrightarrow q, q'_l = 1} \lambda_q$$

is fractional.

Proof
If $\sum_{q \in Q:\ q \leftrightarrow q', q'_l = 1} \lambda_q$ is integer for all $l \in \{1,\ldots,n'\}$, i.e. if the flow on each arc of the underlying network is integer, then, by the flow decomposition theorem, this integer arc flow can be decomposed into integer flows along a finite number of s-t paths, each of which corresponds to a solution $q \in Q$. Hence an integer solution in $\lambda \in I\!N^{|Q|}$ is readily available. ∎

Proposition 6 shows how an underlying network structure can be exploited even when the master is not explicitly formulated in terms of arc flow variables. Desaulniers *et al* (1998) describe many routing applications for which they propose to use a branching scheme based on Proposition 6. Valerio de Carvalho has applied a similar scheme to the cutting stock problem. Even though the subproblem modifications are straightforward, such branching scheme might not be efficient. The difficulty lies in the appropriate selection of an arc on which to branch. There is typically an exponential number of arcs to choose from and a branching constraint that involves a single arc might not be very restrictive. By contrast, a branching constraint based on a component bound, as defined in Proposition 3 and Corollary 5, might amount to bounding the flow a several arcs in the underlying network (for instance, in the standard layered network associated with the dynamic programming solution of the 0-1 knapsack problem, there are i arcs associated with the selection of the ith item). The selection of the underlying network is also important. A network representation that allows for several paths to represent the same solution $q \in Q$ will lead to difficulties inherent to symmetry. Note that, when the column generation subproblem is a resource constrained shortest path problem, the flow decomposition theorem does not hold in the network of the resource constrained problem and Proposition 6 only applies in the larger network associated with the dynamic programming solution of the problem.

3 Implementation

The general branching framework described above includes a wide range of branching rules. The purpose of the present section is: (*i*) to illustrate how the theoretical schemes give rise to simple branching rules which have a practical interpretation; (*ii*) to show that in practise the simplest of these rules that yield limited modifications to the subproblem structure are often sufficient to eliminate all fractional solutions; and (*iii*) to discuss the efficiency of some of these simple rules and, in particular, to emphasize that a branching scheme that yields no subproblem modification might not be the most efficient when, for instance, the solution space exhibits some symmetry. The computer code used to obtain the results reported here is only a preliminary implementation that does not fully exploit the special structure of the subproblem nor does it use heuristics where possible, or optimally implement the separation procedure searching for a branching constraints that cuts

off the current fractional master solution. Nevertheless, this preliminary implementation permits comparison by way of using computational counters. A full-blown computational study of a branch-and-price algorithm for the cutting stock and the bin packing problems is reported in Vanderbeck (1996). The study of a telecommunication network design application by Sutter, Vanderbeck and Wolsey (1998) also contains examples of implementations of our branching scheme.

3.1 The cutting strip problem

The cutting strip problem is quite difficult to solve. For the real instances that we received from Hurkens (1995), the available metal sheets barely suffice to cover the requirements and finding a feasible solution is already very challenging. We use a first fit decreasing heuristic to provide a initial incumbent but it sometimes fails to construct a feasible solution. Moreover, the typical gap between the LP relaxation of the master and the IP solution is significant and solving the problem depends on developing an efficient branching scheme.

Because the master formulation M^{CS} (6) contains convexity constraints, branching can theoretically be straightforward as it can be enforced directly in the subproblems. Indeed, there is a bijective relationship between the master variables and the variables of the original formulation P^{CS} (2), $z_i^k = \sum_{q=(q^k,e^k)\in Q(k)} q_i^k \lambda_q$, and enforcing integrality of each component of z is equivalent to enforcing integrality of the master variables. If $z_i^k \notin I\!N$, let $v = \lceil z_i^k \rceil > z_i^k$ and set $\sum_{q=(q^k,e^k)\in Q(k)} q_i^k \lambda_q \leq v-1$ on one branch and $\sum_{q=(q^k,e^k)\in Q(k)} q_i^k \lambda_q \geq v$ on the other branch. Note that this branching rule is an instance of the branching scheme of Proposition 3, with $B = \{(p+k, \geq, 1),\ (i, \geq, v)\}$, where the component bound $(p+k, \geq, 1)$ restricts q to belong to $Q(k)$. Then, as $\sum_{q\in Q(k)} \lambda_q \leq 1$,

$$\sum_{q\in Q(B)} \lambda_q \geq 1 \quad \Leftrightarrow \quad \sum_{q\in Q(k)} q_i^k \lambda_q \geq v \tag{22}$$

and

$$\sum_{q\in Q(B)} \lambda_q \leq 0 \quad \Leftrightarrow \quad \sum_{q\in Q(k)} q_i^k \lambda_q \leq v-1 \tag{23}$$

Given a fractional solution λ, one can always find a triple (k, i, v) such that $z_i^k =$

$\sum_{q \in Q(k)} q_i^k \lambda_q \notin I\!N$ and $v = \lceil z_i^k \rceil$ on which to branch and the scheme amounts to:

$$\textbf{Rule A:} \qquad \text{enforce} \qquad \sum_{q \in Q(k): q_i^k \geq v} \lambda_q \qquad \in \{0,1\} \qquad \forall k, i, v \ .$$

The implementation of Rule A takes a very simple form. Branching constraint (22) amounts to saying that, in any cutting pattern for sheet k, there must be at least v strips of width w_i. Similarly, branching constraint (23) says that cutting patterns for sheet k can have no more than $v-1$ strips of width w_i. Then, the columns that do not satisfy the branching constraints can be removed from the master formulation and a lower (upper) bound on q_i^k is enforced in the column generation subproblem k. Note that such bounds on q's components can easily be integrated in a special purpose dynamic programming or branch-and-bound algorithm for the knapsack problem, and therefore the tractability of the subproblem is preserved.

This simple branching scheme has been used by Hurkens (1995). However, our computational experiments show that it is not very efficient because of the symmetric structure of the solution space. Although the metal sheet have different dimensions, they can accommodate very similar cutting patterns. Thus, restricting the number of strips of size w_i on a specific sheet k^1, but not on another sheet k^2, can result in an "almost identical" solution where the roles of k^1 and k^2 are interchanged (possibly at the expense of generating new columns). This pitfall must be addressed by an appropriate choice of branching rules. To this end, we tried branching rules that involve all sheet types k.

We first considered a branching scheme based on sets B of component bound constraints of the form $(i, \geq, v)$ only. To find such a set B with minimal cardinality $|B|$ and fractional part ϕ closest to 0.5, where $\phi = (\sum_{q \in Q(B)} \lambda_q) - \lfloor \sum_{q \in Q(B)} \lambda_q \rfloor$, we use the enumerative procedure outlined in Section 2.2. For all our test problems we have always found a single component bound $(i, \geq, v)$ on which to branch and never had to consider a set B with $|B| > 1$. Thus, the branching rule we have found useful is:

$$\textbf{Rule B:} \qquad \text{enforce} \qquad \sum_{q \in Q: q_i \geq v} \lambda_q \qquad \text{integer} \qquad \forall i, v$$

Rule B simply says that the number of cutting patterns containing v or more strips of width w_i must be integer in any integer solution. The implementation of Rule B requires adding a single boolean variable in the column generation subproblem for each branching constraint. Alternatively, if a dynamic program is used to solve the knapsack subproblems, the appropriate modification to the reduced cost can be implemented at the stage where the value of component q_i is fixed.

We observed that if the threshold v is poorly chosen, branching can yield an uneven partition of the solution space (i.e. an unbalanced branch-and-bound tree). Let $q_i^{\max}$ be the maximum value of q_i in any feasible cutting pattern, i.e. $q_i \in [0\,,\, q_i^{\max}]$ for all $q \in Q$. If $q_i^{\max}$ is relatively large, and v is chosen close to the boundary of the interval $[0\,,\, q_i^{\max}]$, then the new master problem on one branch is very restricted, while the other is barely restricted. Say $\sum_{q \in Q: q_i \geq v} \lambda_q = \alpha \notin I\!N$ for $v = q_i^{\max} \gg 0$, then the branching constraint $\sum_{q \in Q: q_i \geq v} \lambda_q \leq \lfloor \alpha \rfloor$ does not significantly improve the LP bound as new columns with entry q_i bounded above by $q_i^{\max} - 1$ instead of $q_i^{\max}$ will be generated. Conversely, if $v = 1 \ll q_i^{\max}$, then the branching constraint $\sum_{q \in Q: q_i \geq v} \lambda_q \geq \lceil \alpha \rceil$ is easily satisfied by incorporating just one unit of i in columns similar to those already generated.

To tackle this issue we tried another way of choosing v that partitions the interval $[0\,,\, q_i^{\max}]$ more evenly. Our numerical experimentations have shown that this could be done by successively considering thresholds $v = \lceil \xi\, q_i^{\max} \rceil$ for $\xi = \frac{1}{2}, \frac{1}{4}, \frac{3}{4}$ and for $i = 1, \ldots, p$ and stopping with the first fraction ξ that yields a fractional sum, $\sum_{q \in Q: q_i \geq \lceil \xi\, q_i^{\max} \rceil} \lambda_q$, for some i. Note that considering larger fraction denominators, i.e. $\xi = \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \ldots$, amounts to slicing the interval in ever smaller portions and thus reproduces the pitfall we were trying to avoid. In our implementation of Rule B, we first try to find a pair (i, v) on which to branch in this way and, if we failed, we resolve on using the enumeration previously described.

Observe that the maximum number of strips of width w_i in a cutting pattern is a function of the sheet type:

$$\max_{q \in Q(k)} q_i^k = \Big\lfloor \frac{W^k}{w_i} \Big\rfloor \quad \text{for } k = 1, \ldots, K.$$

Therefore, when the threshold v is chosen as a fraction of the maximum number of strips,

rule B is an instance of a partitioning of the subproblem space $Q \subseteq \mathbb{N}^{p+K}$ by a hyperplane (Proposition 2), i.e. $(\gamma, \gamma_0) = (e^i, -\lceil \xi \lfloor \frac{W^1}{w_i} \rfloor \rceil, \ldots, -\lceil \xi \lfloor \frac{W^K}{w_i} \rfloor \rceil, 0) \in \mathbb{Z}^{p+K+1}$, where e^i is the i^{th} unit vector of size p.

The other branching scheme we have considered is based on Corollary 5. Indeed, according to Martello and Toth (1990), the most effective way to solve bounded knapsack problems nowadays is to transform it into its 0-1 form and to apply an efficient algorithm designed for the 0-1 knapsack. In the case of the cutting strip problem, the column generation subproblem is an unbounded knapsack problem but the implicit component bounds that result from the knapsack constraint are reasonably small for the practical instances we considered. Therefore, the number of binary variables in the 0-1 form of the knapsack subproblem remains practical. If the column generation subproblem is solved in its 0-1 form, branching on the components of the associated binary solution vectors is most appropriate.

Hence, the branching rule we have implemented is the following (the notations we used here are the same as in Corollary 5),

$$\textbf{Rule C}\text{: enforce} \qquad \sum_{q \in Q : q \leftrightarrow q' \,\&\, q'_l = 1} \lambda_q \qquad \text{integer} \qquad \forall l \in \{1, \ldots, n'\}$$

Given a fractional solution λ, we search for a index l, i.e. a component of the 0-1 form of columns q, such that the number of columns with an entry one in that component, $\sum_{q \in Q : q \leftrightarrow q' \,\&\, q'_l = 1} \lambda_q$, is fractional. To avoid the pitfall of an unbalanced tree as described above, we consider indices $l = p_i + n_i - 1$ for each width i before considering $l = p_i + n_i - 2$ and so forth. When the 0-1 form of the column generation subproblem is used, the modifications that results from branching rule C merely consist in amending the objective coefficients of the subproblem binary variables for the associated indices l.

Note that, after applying rule B or C, there might remain fractional solutions where the fractional columns correspond to identical cutting patterns but on different sheet types k. Therefore Rule A is still used as a complement of rule B and the subsystem level equivalent of Rule C (enforcing $\sum_{q \in Q(k) : q \leftrightarrow q' \,\&\, q'_l = 1} \lambda_q$ integer $\forall k,\ l \in \{1, \ldots, n'\}$) is used

to complement Rule C.

3.2 The cutting stock problem

With his specific cost structure, the cutting stock problem is known to have a strong master LP relaxation: for most instances, rounding up the master LP solution provides the IP value, i.e. typically $Z_{IP} = \lceil Z_{LP}^{root} \rceil$ (Marcotte 1985). Hence, branching essentially aims at proving the optimality of the root node bound by obtaining a incumbent solution of the same cost. In this context, primal heuristics can be quite helpful (see Vanderbeck, 1996). However, as our purpose here is to test branching schemes, we did not incorporate heuristics other than a straightforward first fit decreasing heuristic (Martello and Toth, 1990) that is used for the sake of comparison with Valerio de Carvalho who also uses this heuristic. When the incumbent solution obtain at the root node has a cost strictly higher than $\lceil Z_{LP}^{root} \rceil$, branching takes place. Prior to branching however, we enforce integrality of the cost by adding the cut $\sum_{q \in Q} \lambda_q \geq \lceil Z_{LP}^{root} \rceil$ to the master formulation if needed. This constraint has the same form as a branching constraint. The resulting subproblem modifications simply consist of subtracting from the reduced cost a constant equal to the corresponding dual variable.

A branching scheme for the cutting stock problem has been proposed by Vance (1996). Vance generates only maximal cutting patterns and branches on a single fractional column, say $\hat{q}$. The branch $\lambda_{\hat{q}} \geq \lceil \alpha \rceil$, is dealt with by adjusting the master RHS appropriately. In the branch $\lambda_{\hat{q}} \leq \lfloor \alpha \rfloor$, the component bounds in the subproblem are amended to avoid regenerating $\hat{q}$. This unbalanced branching scheme is used in a depth-first-search approach, in an attempt to construct a feasible integer solution quickly. The scheme works well, with few exceptions (odd instances for which there is significant backtracking). Our limited computational tests show that applying a less specific branching scheme such as rules B or C works just as well and is more robust. Although rules B or C are only partial branching schemes not guaranteed to yield an integer solution, they have been sufficient in solving all test problems obtained from Vance to optimality.

In Valerio de Carvalho 's study (1996) of the cutting stock problem, the master is formulated in terms of arc flow variables and branching consists in enforcing arc flow inte-

grality. The test problems used by Valerio de Carvalho are from the OR-Library (Beasley 1990). We have solved some of these instances using Rule C. However for these instances, the feasible cutting patterns involve only a few (1 or 2) strips of any given width (i.e. n' is barely larger than n) and it has not always been possible to eliminate all fractional solutions by branching on a single component of q'. Therefore we considered branching on pairs of components.

The branching rule we have implemented is:

$$\textbf{Rule D}\text{: enforce} \quad \sum_{q \in Q: q \leftrightarrow q' \& q'_l = q'_k = 1} \lambda_q \quad \text{integer} \qquad \text{for } l,\ k \in \{1, \ldots, n'\}$$

which is another instantiation of the branching scheme of Corollary 5. To account for the reduced cost modifications that result from a branching constraint based on Rule D, we add an extra binary variable $x_{l\,k}$ in the 0-1 knapsack formulation whose weight is $w_{l\,k} = w_l + w_k$ and whose objective coefficient is equal to the sum of those variables x_l and x_k plus the dual variable associated with the branching constraint, and we enforce the constraint $x_l + x_k + x_{l\,k} \leq 1$. Such GUB constraints can easily be handled in a specialized branch-and-bound procedure for the 0-1 knapsack as long as they concern disjoint subsets of variables. Thus, in applying Rule D, we consider only components k and l that are not in any other Rule D type branching constraints at the current node. The combination of Rules C and D (with restricted selection of component pairs) has enabled us to solve all the test problems to optimality.

3.3 Computational results

For our computational tests, we used 22 cutting stock problem instances and 17 cutting strip instances. The first 10 data sets are real-life cutting stock instances used by Vance in her study (1996). We have selected the 10 instances for which Vance's algorithm performed the largest total number of master iterations. They are named "npx", where n is the number of different widths and x is the problem number used by Vance. For these instances the average width item can fit between 4 to 15 times on a roll. The following 12 instances are bin packing instances from the OR-Library (Beasley 1990). As these instances involve multiple items of the same size, they are in fact integer cutting stock

problems. Our selection of instances is based on Valerio de Carvalho 's computational tests: for each problem size we kept the two instances for which Valerio de Carvalho 's algorithm required the largest number of branch-and-bound nodes. For these instances the feasible cutting patterns include between 1 to 5 strips. The instance names are the OR-Library reference names, followed by the number of different item widths.

The next 7 instances have been generated by hand drawing from real-life cutting strip instances used by Hurkens (1995). Each instance is referred to by a name "nbK" where n is the number of widths and K the number of roll types. The last 10 instances of the cutting strip problem have been generated randomly using uniform discrete distributions with parameters in the same range as that of the real-life instances: we generated d_i (cfr P^{CS} (2)) from $U(400, 2000)$, w_i from $U(50, 300)$, W^k from $U(900, 1000)$, L^k from $U(180, 220)$, and rolls are generated as long as the total metal surface already available is less than 105% of the required surface. These instances are named "nrK". Instances of the same dimensions are distinguished by a trailing letter.

Over the years we have developed a modular implementation of an IP column generation algorithm that can be easily adapted to new applications (Vanderbeck 1994). The code, called IPCG, is written in C and uses CPLEX 3.0 to solve the master LPs. The column generation subproblems are solved using CPLEXMIP, unless a special purpose algorithm is implemented. Here, we use the algorithm MTR of Martello and Toth (1990) to solve the 0-1 form of the knapsack subproblems, except for the OR-Library instances for which we use branching rule D and which have been solved using our own implementation of a branch-and-bound code for the 0-1 knapsack (based on the presentation found in Nemhauser and Wolsey, 1988, p.455). For the tests reported here, we have not implemented any specific algorithm for the general integer knapsack but we solve the subproblems with CPLEXMIP when branching rules A and B are used. Therefore, the computational times cannot be used to compare the performance of the different branching rules and we use counters instead. The computations have been carried out on a HP9000/712/80 workstation (64Mb of main memory).

In Tables 1 and 2, we present our computational results for the cutting stock and the cutting strip problem respectively. These results were obtained using branching Rule C

(in combination with Rule D for the OR-Library instances) which seems to perform best overall. The columns of the tables contain the problem names, the lower bound obtained at the root node (RB), the IP solution, the initial heuristic solution (HR), the branch-and-bound tree depth (dep), the number of nodes processed in the branch-and-bound tree (nod), the number of master LP solved (mast), the total number of columns generated (col), and the CPU time in seconds (time).

Tables 1, 3, 4 and 5 contain comparative results. The performance of different branching schemes are assessed using measures of the branch-and-bound tree size and counters of the number of master iterations and column generations. Table 3 shows that Rule B has an efficiency comparable to that of Rule C. When using a general MIP code to solve the subproblems with branching rules A and B, we experienced some difficulties (rounding problems, time overruns) in solving some instances which therefore have not been included in the tables. Table 4 shows that, for the cutting stock problem, rule C performs just as well or better than the branching scheme of Vance. As Vance does not construct a heuristic solution at the outset, we did not use the first fit decreasing (FFD) heuristic for this test and Rule C proved to be good at building a primal solution (the comparison can be made with the results of Table 1). In the last column of Table 1 (VdC nod), we have included the number of branch-and-bound tree nodes obtained by Valerio de Carvalho who is branching on the arc flow variables. The comparison with our number of branch-and-bound tree node shows that, for cutting stock instances from the OR-Library, the combination of rules C and D tends to yield smaller branch-and-bound tree than that obtained by Valerio de Carvalho using a branching rule based on Proposition 6. Table 5 shows that using global branching constraints such as Rule B is very helpful in dealing with a symmetric solution space such as that of the cutting strip problem. Here, instances 4b7c and 4b7 could not be solved with Rule A due to lack of computer memory. Instance 4b7c was solved by Hurkens (1995). Using Rule A, he obtained a branch-and-bound tree of depth 39 with 7189 nodes and he solved 7374 master LPs.

name	RB = IP	HR	dep	nod	mast	col	time	VdC nod
11p4	101	103	39	45	99	80	2.67	
5p11	21	21	0	1	1	10	0.12	
14p12	56	57	23	24	65	62	1.47	
7p14	70	71	3	4	11	20	0.26	
7p17	9	9	0	1	2	10	0.17	
7p18	91	94	6	8	23	29	0.45	
12p19	23	24	8	11	41	48	0.79	
6p20	13	13	0	1	3	14	0.14	
18p22	33	33	0	1	38	58	0.90	
12p24	58	60	21	24	73	67	1.41	
u250_06_78	102	103	43	44	261	275	25.8	44
u250_12_76	105	107	8	14	162	207	5.25	52
u500_08_81	196	199	34	44	189	227	8.54	76
u500_15_80	201	204	27	31	188	240	7.57	67
u1000_03_81	411	416	37	41	141	193	8.53	89
u1000_14_81	394	421	58	65	186	221	14.3	135
t120_04_92	40	46	14	15	402	434	39.1	31
t120_19_87	40	46	15	16	375	406	38.9	32
t249_15_142	83	95	48	50	664	710	387	70
t249_17_145	83	94	61	69	813	838	542	73
t501_03_199	167	190	135	150	1377	1403	3312	134
t501_18_193	167	189	105	114	949	1008	1671	131

Table 1: Computational results for the cutting stock problem (using rules C and D) and comparison with the number of branch-and-bound nodes obtained by Valerio de Carvalho, using the branching scheme of Proposition 6, for cutting stock instances from the OR-Library.

name	RB	IP	HR	dep	nod	mast	col	time
3b2	2	3	3	4	15	26	26	0.44
3b4	0	330	886	5	17	45	24	0.65
4b7b	0	0	60	3	4	8	45	0.23
4b7e	188890	433274	3312331	11	89	131	185	4.50
4b7c	702054	1674031	infeas	21	545	843	331	33.27
4b7	1054	1462	1504	21	639	1323	689	66.81
4b7d	8804	26979	178501	22	961	1465	471	74.85
4r4	583	920	23471	9	33	52	61	1.10
4r3b	0	380	6993	7	35	55	54	0.97
4r6c	8436	10056	45271	8	45	63	80	1.30
5r7	3057	4367	38140	22	1085	1327	269	58.07
5r6b	147	963	35255	14	291	339	144	9.85
5r6c	1682	4048	30587	24	2531	3011	370	161.44
6r7	0	0	infeas	19	69	108	139	3.43
6r10b	1562	2590	62584	27	1843	1986	277	125.08
6r8c	0	752	40320	28	5903	6528	396	514.00
7r10	0	200	28658	26	2171	2378	311	185.15

Table 2: Computational results for the cutting strip problem (using rule C and its subsystem equivalent).

	Rule B (and A)				Rule C (and complement)			
name	dep	nod	mast	col	dep	nod	mast	col
5p11	0	1	1	10	0	1	1	10
7p14	18	19	26	20	3	4	11	20
7p17	0	1	2	10	0	1	2	10
7p18	7	8	27	33	6	8	23	29
6p20	0	1	3	14	0	1	3	14
18p22	0	1	43	63	0	1	38	58
3b2	5	13	22	24	4	15	26	26
3b4	6	17	36	23	5	17	45	24
4b7b	3	4	8	45	3	4	8	45
4b7e	14	79	123	174	11	89	131	185
4b7c	24	1415	1765	508	21	545	843	331
4b7	24	871	1431	616	21	639	1323	689
4r4	6	35	50	57	9	33	52	61
4r3b	8	29	49	52	7	35	55	54
4r6c	10	79	115	92	8	45	63	80
5r7	21	975	1126	225	22	1085	1327	269
5r6b	13	205	234	114	14	291	339	144
5r6c	21	1073	1283	232	24	2531	3011	370
6r7	5	18	30	76	19	69	108	139

Table 3: Comparing branching rules B and C for cutting stock and cutting strip problems.

	Vance's Branching			Rule C & no FFD heur		
name	dep	nod	mast	dep	nod	mast
11p4	605	1213	3753	26	31	101
5p11	40	71	143	4	5	13
14p12	60	99	341	25	31	99
7p14	41	66	118	3	4	11
7p17	33	49	108	25	36	113
7p18	71	89	135	11	14	46
12p19	26	39	150	24	32	93
6p20	29	55	136	6	7	23
18p22	47	74	224	47	64	225
12p24	26	38	135	8	9	53

Table 4: Comparing the branching scheme of Vance to branching rule C for the cutting stock problem, with no heuristic solution at the outset.

4 Further Remarks

We emphasize that some of the branching rules we used would not be easy to formulate if IP decomposition was based on the convexification of the subsystem instead of on its discretization. Consider using rule B type branching constraints for the cutting strip problem in a master formulation resulting from subsystem convexification. For the sake of argument, assume that the current fractional solution uses $\alpha \notin I\!N$ cutting patterns with at least v strips of width w_i. Under the convexification approach, the master variable are not directly restricted to be integer and the branching constraint $\sum_{q \in Q: q_i \geq v} \lambda_q \geq \lceil \alpha \rceil$ is not valid since a cutting pattern z^k could be defined as $\frac{1}{2} q^k$ for instance. Enforcing the corresponding branching constraint requires adding K binary variables to the master, $s^k = s^k(\lambda)$ for $k = 1, \ldots, K$, where $s^k = 1$ if $z_i^k = \sum_{q \in Q(k)} q_i \lambda_q \geq v$ and zero otherwise and the relations $s^k = s^k(\lambda)$ take a form similar to that used in our column generation subproblem modification scheme:

$$v\, s^k \leq \sum_{q \in Q(k)} q_i \lambda_q \leq (v-1) + (\lfloor \frac{W^k}{w_i} \rfloor - v + 1)\, s^k \quad \text{for } k = 1, \ldots, K\ .$$

	Rule A				Rule B, then A			
name	dep	nod	mast	col	dep	nod	mast	col
3b2	3	9	16	22	5	13	22	24
3b4	6	21	38	22	6	17	36	23
4b7b	17	18	27	66	3	4	8	45
4b7e	22	343	417	145	14	79	123	174
4b7c	>19	>4603	>6732		24	1415	1765	508
4b7	>28	>4594	>6445		24	871	1431	616
4r4	5	17	24	35	6	35	50	57
4r3b	6	21	35	34	8	29	49	52
4r6c	17	493	638	80	10	79	115	92
5r7	23	2542	2732	201	21	975	1126	225
5r6b	16	286	331	114	13	205	234	114
5r6c	22	6475	6734	269	21	1073	1283	232
6r7	10	22	35	76	5	18	30	76

Table 5: Comparing the use of the sole branching rule A to the use of branching rule B complemented by A for the cutting strip problem.

Then, the constraint

$$\sum_{k=1}^{k} s^k \geq \lceil \alpha \rceil$$

is added to the master and up to K branching stages might be required to force each s^k to its integer value. This remark points out that the modifications to the column generation subproblem that arise from our branching schemes are not unduly cumbersome because implementing such branching constraints would be even more complex in the convexification approach. Moreover, in cases such as that of the cutting stock problem, where the transformation from the master variables to the original variables is not straightforward, the above attempt to implement such branching constraints in terms of the original variables falls short.

In a decomposition based on subproblem discretization, one may have to generate columns $q \in Q$ that are interior points of $conv(X)$. Such interior points will arise as solutions to the modified column generation subproblem. Indeed, adding auxiliary variables in the subproblem amounts to working in a space of higher dimension, $X' \subseteq I\!N^n \times \{0,1\}^{|G^u|+|H^u|}$. The subproblem solution will be an extreme point of X' whose projection in X can be in the interior of $conv(X)$. Typically, only a small subset of interior points of $conv(X)$ will be considered in the course of the column generation procedure.

The branching constraints that we have introduced amount to defining an additional integer variable

$$s = \sum_{q \in Q} \gamma(q) \lambda_q$$

where the coefficients $\gamma(q)$ are functions of the vectors $q \in Q$, and enforcing its integrality, i.e. if $s = \alpha \notin I\!N$, branch on $s \leq \lfloor \alpha \rfloor$ or $s \geq \lceil \alpha \rceil$. This interpretation suggests that in designing a branching scheme one should not be restricted to branching on the existing variables. Moreover, the newly defined variables often have a natural signification for the problem on hand. In the branching schemes considered in Section 2, $\gamma(q)$ are binary functions. The class of newly defined variables of which to branch could be widen to include cases where functions $\gamma(q)$ are not restricted to 0-1 values.

Observe that the introduction of cutting planes in the master will require subproblem modifications that are similar to those required by branching constraints. As we have

seen, an additional master constraint of the form

$$\sum_{q \in Q} \gamma(q) \lambda_q \leq \gamma_0$$

can be handled at the subproblem level as long as the relations $\gamma(q)$ that define the constraint coefficients can be expressed as MIPs. Moreover, for special value of the constraint coefficients, the modification scheme simplifies. For instance, if $\gamma(q) \geq 0 \ \forall q \in Q$ and $\gamma_0 = 0$, then the constraint can be directly enforced in the subproblem by adding the constraint $\gamma(q) = 0$ in the subproblem formulation. When $\gamma(q)$ is a linear function of q, the modifications simply amount to amending the subproblem objective coefficients. We have seen cases where $\gamma(q)$ was not a linear function of q, but could be expressed as a linear function of q', a dissagregation of q (for instance, its 0-1 form, or its associated arc flow incidence vector in some underlying network).

The issue of branching efficiency touched on in this paper is clearly not specific to IP decomposition. The study of branching constraints that yield much improvement in the bounds on every branch, deal effectively with a symmetric solution space and / or produce good primal solution quickly merits further research. To our knowledge, the literature on these questions is mostly limited to application specific computational studies. In a column generation framework, an additional issue is to be considered: the tradeoff between branching efficiency and its implementation complexity in terms of the required modifications to the subproblem. This paper and that of Vance (1996) explicitly examine this tradeoff by way of comparative computational tests.

Acknowledgement

We are grateful to Cor Hurkens and Pamela Vance for providing information and data for the cutting strip and the cutting stock problem respectively, and to Jacques Desrosiers and Laurence Wolsey for their constructive comments. We especially want to thank the anonymous referees who provided a very detailed and very helpful feedback. This research was supported in part by a Management Research Fellowship granted by the British Economic and Social Research Council (ESRC).

References

Anbil R., C. Barnhart, L. Hatay, E.L. Johnson, and V.S. Ramakrishnan (1993). Crew-Pairing Optimization at American Airlines Decision Technologies, *Optimization in Industry*, T.A. Cirani and R.C. Leachman eds., Wiley, 31-36.

Barnhart C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance (1994). Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Mathematical Programming, State of the Art*, book of the International Symposium on Mathematical Programming, 1994, edited by J.R.Birge and K.G. Murty.

Beasley J.E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research society*, **41**, pp 1069-1072.

Using the CPLEX Linear Optimizer, version 3.0. (1994). CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd., Bldg. 802, Incline Village, NV 89451-9436. (702) 831-7744.

Dantzig G.B. and P. Wolfe (1960). Decomposition Principle for Linear Programs, *Operations Research*, **8**, 101-111.

Desaulniers G., J. Desrosiers, I. Ioachim, M.M. Solomon, and F.Soumis (1998). A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems, in *Fleet Management and Logistics*, T.G. Crainic and G. Laporte (eds), Kluwer, Boston, MA, 57-93.

Desrosiers J., Y. Dumas, M.M. Solomon and F. Soumis (1994). Time Constrained Routing and Scheduling, Chapter 2 in M.E. Ball, T.L. Magnanti, C. Monma, and G.L. Nemhauser eds *Handbooks in Operations Research and Management Sciences: Networks*, North-Holland.

Gilmore P.C. and R.E. Gomory (1961), A Linear Programming Approach to the Cutting Stock Problem, *Operations Research*, **9**, 849-859.

Gilmore P.C. and R.E. Gomory (1963), A Linear Programming Approach to the Cutting Stock Problem: Part II, *Operations Research*, **11**, 863-888.

Geoffrion A.M. (1974), Lagrangian Relaxation for Integer Programming, *Mathematical*

Programming Study 2, 82-114.

Hurkens C. (1995). Private Communications.

Johnson E.L., A. Mehrotra and G.L. Nemhauser (1993). Min-Cut Clustering, Mathematical Programming, 62, 1993, pp 133-151.

Marcotte O. (1985). The Cutting Stock Problem and Integer Rounding, *Mathematical Programming*, **33**, 89-92.

Martello S. and P. Toth (1990). *Knapsack Problems: Algorithms and computer Implementations*, Wiley-Interscience Series in Discrete Mathemathics and Optimization.

Martin R. K. (1987). Generating Alternative Mixed-Integer Programming Models Using Variable Redefinition. *Operations Research*, Vol. **35**, No. 6, pp 820-831.

Martin R. K. (1991), Using Separation Algorithms to generate Mixed Integer Model Formulations. *Operations Research Letters*, **10**, pp 119-128.

Nemhauser G.L. and L.A. Wolsey (1988). *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc.

Savelsbergh, M.W.P. (1997). A Branch-and-Price Algorithm for the Generalized Assignment Problem. Operations Research 45, 831-841.

Sutter A. , F. Vanderbeck and L.A. Wolsey (1998). Optimal Placement of Add/Drop Multiplexers: Heuristic and Exact Algorithms, *Operations Research*, Vol. **46**, No. 5, pp 719-728.

Valerio de Carvalho J.M. (1996). Exact Solution of bin-packing problems using column generation and branch-and-bound. *Working paper. Depart. Prouducao e Sistemas*, Universidade do Minho, 4709 Braga Codex, Portugal.

Vance P.H., C. Barnhard, E.L. Johnson, and G.L. Nemhauser (1994). Solving Binary Cutting Stock Problems by Column Generation and Branch-and-bound. *Computational Optimization and Applications* **3**, p. 111-130.

Vance, P.H. (1996), Branch-and-Price Algorithms for the One-Dimensional Cutting Stock

Problem. Working paper. Department of Industrial Engineering, Auburn University, Auburn, Alabama 36849-5346. To appear in *Computational Optimization and Applications*.

Vanderbeck F. (1994). Decomposition and Column Generation for Integer Programs, *Ph.D. Thesis*, Faculté des Sciences Appliqées, Université Catholique de Louvain, Louvain-la-Neuve.

Vanderbeck F. (1996). Computational Study of a Column Generation algorithm for Bin Packing and Cutting Stock problems, *Research Papers in Management Studies*, University of Cambridge, 1996 no 14, forthcoming in Mathematical Programming.

Vanderbeck F. and L. A. Wolsey (1996). An Exact Algorithm for IP Column Generation, *Operations Research Letters* Vol. 19, No. 4, pp 151-159.