***Les Cahiers du GERAD*** ISSN: 0711–2440

# Cutting Stock Problems

H. Ben Amor
J.M. Valério de Carvalho

G–2004–30

April 2004

# Cutting Stock Problems

**H. Ben Amor**
*GERAD and Département de mathématiques et génie industriel*
*École Polytechnique*
*C.P.6079, succ. Centre-ville, Montréal, H3C 3A7 Canada*
Hatem.Ben.Amor@gerad.ca

**J.M. Valério de Carvalho**
*Dept. Produção e Sistemas*
*Universidade do Minho*
*4710-057 Braga, Portugal*
vc@dps.uminho.pt

April, 2004

*Les Cahiers du GERAD*

G–2004–30

**Abstract**

Linear relaxations are solved by column generation. Stabilization techniques such as dual-optimal inequalities and stabilized column generation algorithms that have been proposed to improve the efficiency of this process are briefly discussed. Integer solutions are obtained by combining heuristics and branch-and-price schemes. We survey the basic models proposed for cutting stock and the corresponding solution approaches. Extended Dantzig-Wolfe decomposition is surveyed and applied to these models in order to show the links to Gilmore–Gomory model. Branching schemes discussion is based on the subproblem formulation corresponding to each model.

**Résumé**

La génération de colonnes fut proposée par Gilmore et Gomory pour la résolution du problème de découpe unidimensionnelle, indépendemment de la décomposition de Dantzig-Wolfe. Les relaxations linéaires sont résolues par génération de colonnes. Plusieurs techniques de stabilisation, telles que les inégalités dual-optimales et la génération de colonnes stabilisée, qui ont été utilisées pour l'accélération de ce processus sont discutées brièvement. La résolution en nombres entiers est effectuée en combinant des heuristiques primales avec un schéma de branch-and-price. Nous présentons les différents modèles proposés et les approches de résolutions associées. Des extensions de la décomposition de Dantzig-Wolfe aux problèmes à variables entières sont appliquées aà ces modèles pour montrer les liens avec la formulation de Gilmore et Gomory. Pour chaque modèle, des schémas de branchement basés sur la formulation du sous-problème correspondant sont discutés.

**Acknowledgments:** The second author was partially supported by FCT – Fundação para a Ciência e a Tecnologia (Projecto POSI/ 1999/ SRI/ 35568) and by Centro de Investigação Algoritmi da Universidade do Minho (Grupo de Engenharia Industrial e de Sistemas).

## 1 Introduction

The cutting stock problem (CSP) was one of the problems identified by Kantorovich in his paper entitled "Mathematical methods of organizing and planning production", that first appeared in 1939, in Russian, and was later published in Management Science (1960). The problem consist of determining the best way of cutting a set of large objects into smaller items. There are large potential economic savings resulting from the solution of this kind of problems. CSP are encountered in a wide variety of industrial applications, such as in the steel, wood, glass and paper industries, and also in service sector applications, such as cargo loading and logistics.

In this paper, we focus on one-dimensional problems. Since Gilmore and Gomory proposed the use of column generation (CG) to solve its linear programming (LP) relaxation (Gilmore and Gomory (1961, 1963)), several solution approaches for this problem were based on algorithms using column generation complemented by heuristics. The nineties mark a turning point in this field: several algorithms combining column generation and branch and bound were proposed to solve the CSP. It was also recognized that those algorithms were also useful to solve instances with many items of many different sizes, yielding a low average demand, a problem that is usually denoted as the bin packing problem (BPP) or the Binary Cutting Stock Problem (BCSP). In this problem, items are assigned to bins in such a way that the capacity of the bins is not exceeded and the number of bins is minimized.

The one-dimensional CSP and BPP are essentially the same problem, even though, under Dyckhoff's system (Dyckhoff (1990)), they were classified as 1/V/I/R and 1/V/I/M, respectively. The reason that possibly motivated the use of a different classification for them was that different solution methods had been traditionally used to address them.

Literature reviews often include not only the CSP and the BPP, but also other problems closely related to them, as knapsack, vehicle loading and pallet loading problems, as well as many others. Examples are Sweeney and Paternoster (1992) and Dowsland and Dowsland (1992). The book by Dyckhoff and Finke (1992) identifies more than 700 papers on Cutting and Packing, and classifies them. An annotated bibliography was proposed by Dyckhoff et al. (1997). Many papers refer to case studies where column generation is used to get solutions for real world applications in the aluminium industry (Stadtler (1990); Helmberg (1995)), in the steel industry (Valério de Carvalho and Guimarães Rodrigues (1995)), in the paper industry (Goulimis (1990)), and in the forest industry (Sessions and Guanda (1988), Sessions and Garland (1989)).

This paper is organized as follows. In Section 2, we review the cutting stock models introduced by Kantorovich and by Gilmore and Gomory, a model based on arc-flows, and an acyclic network based vehicle routing problem (VRP) model. We also comment on the quality of the bounds that result from their linear programming relaxations. Models with strong linear programming relaxations are of crucial importance in solving integer programming problems. The Dantzig-Wolfe decomposition is a tool that can be used to obtain stronger models when extended to integer programming problems. We present some concepts from this method and its application to Cutting Stock. In Section 3, we address the integer solution of cutting stock problems, using heuristics and column generation combined with branch and bound. We review the main issues, several branching schemes, and comment on computational results. In Section 4, we show that stabilization techniques can improve the behaviour of column generation for solving the linear programming relaxation of the Gilmore-Gomory model. In Section 5, we present some extensions of the one-dimensional cutting stock problem and, in Section 6, some directions for future research.

## 2 Mathematical programming Models

The one-dimensional CSP consists of determining the smallest number of rolls of width $W$ that have to be cut in order to satisfy the demand of $m$ clients with orders of $b_i$ rolls of width $w_i, i = 1, 2, \ldots, m$. $W$, $w_i$ and $b_i$ $(i = 1, \ldots, m)$ are assumed to be positive integers.

### 2.1 Kantorovich model

Kantorovich (1960) introduced the following mathematical programming model for the CSP to minimize the number of rolls used to cut all the items:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{K} x_0^k && (1)\\
\textit{subj. to} \quad & \sum_{k=1}^{K} x_i^k \geq b_i,\ i = 1, \ldots, m && (2)\\
& \sum_{i=1}^{m} w_i x_i^k \leq W x_0^k,\ k = 1, \ldots, K && (3)\\
& x_0^k = 0 \text{ or } 1,\ k = 1, \ldots, K && (4)\\
& x_i^k \geq 0 \text{ and integer},\ i = 1, \ldots, m, \quad k = 1, \ldots, K && (5)
\end{aligned}$$

where $K$ is a known upper bound on the number of rolls needed, $x_0^k = 1$, if roll $k$ is used, and 0, otherwise, and $x_i^k$ is the number of times item $i$ is cut in roll $k$. Constraints (2) enforce the satisfaction of the demand for the items, and constraints (3) guarantee that the items cut in a roll do not exceed its capacity. The latter group is usually denoted as the knapsack constraints. Indeed, when $x_0^k = 1$, (3) is exactly a knapsack constraint with capacity $W$; and when $x_0^k = 0$, (3) is a knapsack constraint with capacity 0 (i.e. all variables $x_i^k$ equal 0 and the $k$th bin is not used).

A lower bound for the integer optimum can be obtained from the solution of the linear programming relaxation, which results from substituting the two last constraints for $0 \leq x_0^k \leq 1$ and $x_i^k \geq 0$, respectively. This bound can be very weak. It is equal to the minimum amount of space that is necessary to accommodate all the items, $\lceil \sum_{i=1}^{m} b_i w_i / W \rceil$, and can be very poor for instances with large waste. In the limit, as $W$ increases and all the items have a size $w_i = \lfloor W/2+1 \rfloor$, the integer optimal value is $\sum_{i=1}^m b_i w_i$ whereas the lower bound approaches $(1/2)\sum_{i=1}^m b_i w_i$ (Martello and Toth (1990)). This is a drawback of the model. However, computational experiments show that the most difficult instances are those with very small loss (i.e. their linear relaxation optimal value is very difficult to obtain) for whom the lower bound is equal to optimal integer objective. Furthermore, its solution space has symmetry. Different solutions to the model, with the same cutting patterns swapped in different rolls, will correspond to the same global cutting solution. This means that any efficient branching has to make decisions independently from $k$. Branching directly on individual $x_0^k-$ variables or $x_i^k-$ variables will not eliminate the current fractional solution.

## 2.2 Gilmore-Gomory model

Gilmore-Gomory proposed a model in which the possible cutting patterns are described by the vector $A^p = (a_1^p, \ldots, a_i^p, \ldots, a_m^p)^T$, where the element $a_i^p$ represents the number of items of width $w_i$ obtained in cutting pattern $p$. A cutting pattern $p$ is valid if

$$\sum_{i=1}^{m} a_i^p w_i \le W \tag{6}$$

$$a_i^p \ge 0 \text{ and integer .} \tag{7}$$

Define $P$ as the set of all feasible patterns and let $\lambda^p$ be a decision variable that denotes the number of rolls cut according to cutting pattern $p$, for all $p \in P$. The CSP is modelled as follows:

$$\min \quad \sum_{p\in P} \lambda^p \tag{8}$$

$$subj.\ to \quad \sum_{p\in P} a_i^p \lambda^p \ge b_i,\ \ i = 1, 2, \ldots, m \tag{9}$$

$$\lambda^p \ge 0 \text{ and integer, } \forall p \in P. \tag{10}$$

The number of columns in formulation (8)–(10) may be very large even for moderately sized problems. As it is usually impractical to enumerate all the columns, Gilmore and Gomory proposed column generation to solve its LP relaxation (Gilmore and Gomory (1961)). The problem is initialized with a set of cutting patterns (for instance, each one with multiple copies of the same item in quantities $\lfloor \frac{W}{w_i} \rfloor, \forall i$), and the dual information is used to price the columns out of the master problem. Let $\pi = (\pi_1, \ldots, \pi_m)$ be the dual variables associated to the constraints (9), and $\bar{\pi}$ the dual optimal solution at a given column generation iteration. The "most attractive" column is given by the solution of the following knapsack problem:

$$max \quad \sum_{i=1}^{m} \bar{\pi}_i a_i \tag{11}$$

$$\sum_{i=1}^{m} a_i w_i \leq W \tag{12}$$

$$a_i \geq 0 \text{ and integer,} \tag{13}$$

which corresponds to finding the column $A^{min}$ with the minimum reduced cost $\bar{c}_{min} = 1 - \bar{\pi} A^{min} = min_{p \in P}(1 - \bar{\pi} A^p)$. If the reduced cost is negative, the column is added to the restricted master problem, which is re-optimized; otherwise, the solution is optimal (to the linear relaxation).

A lower bound can be easily calculated at any iteration, using duality. In matrix form, the dual of the CSP is $\max\{\pi b : \pi A^p \leq 1, \pi \geq 0\}$. As seen, $1 - \bar{\pi} A^{min} \leq 1 - \bar{\pi} A^p, \forall p \in P$, which is equivalent to $(\bar{\pi}/\bar{\pi} A^{min}) A^p \leq 1, \forall p \in P$. This means that $(\bar{\pi}/\bar{\pi} A^{min})$ is a feasible solution to the dual of the CSP (with all valid columns enumerated). The value of this feasible dual solution, $(\bar{\pi}/\bar{\pi} A^{min}) b$, is a lower bound to the value of the primal problem, and is equal to the value of the optimal current solution, $\bar{\pi} b$, divided by the optimal value of the knapsack subproblem, $\bar{\pi} A^{min}$ (Farley (1990)). This bound can be used to cut-off the tails of column generation processes. However, even if this bound is better than the lagrangean bound $b^T \pi + \bar{c}_{min} \sum_{p \in P} \lambda_p$ (where $\bar{c}_{min} = 1 - \bar{\pi} A^{min}$ is the minimum reduced cost computed at some CG iteration), it does not cut more than one iteration in practice (Ben Amor (1997)). This may be explained by the following. Farley's lower bound can be written as $\bar{z}/(1 - \bar{c}_{min})$ where $\bar{z}$ is the optimal value of the restricted master problem at the current CG iteration (it can also be obtained by substituting the optimal value of the linear relaxation of CSP $z_{lp}$ to $\sum_{p \in P} \lambda_p$ in the lagrangean bound expression above, see Ben Amor (1997)). In order to have $\bar{z}/(1 - \bar{c}_{min})$ close to $\bar{z}$, $\bar{c}_{min}$ should be close to 0 which usually happens during the very last CG iterations. This lower bound may be improved by considering items that need to be cut in different patterns separately (see Ben Amor (1997) for details).

The bound given by the LP relaxation of Gilmore-Gomory's model is known to be very tight. Most of the one-dimensional cutting stock instances have gaps smaller than one, and we say that the instance has the integer round-up property, but there are instance with gaps equal to 1 (Marcotte (1985, 1986)), and as large as $\frac{7}{6}$ (Rietz and Scheithauer (2002)). It has been conjectured that all instances have gaps are smaller than 2, a property denoted as the modified integer round-up property (Scheithauer and Terno (1995)).

### 2.3 Extended Dantzig-Wolfe decomposition for integer programs

Dantzig-Wolfe decomposition (Dantzig and Wolfe (1960)) was initially designed for structured linear programming problem. The resulting master problem is well suited for column generation. When extended to integer programming (IP) (see Vanderbeck (2000b); Ben Amor (2002)), it becomes a powerful tool to obtain models with stronger LP relaxations for combinatorial optimization problems. Consider the integer programming problem, assumed to be feasible,

$$\begin{aligned}
\min \quad & cx && (14)\\
\textit{subj. to} \quad & Ax = b && (15)\\
& x \in X && (16)\\
& x \geq 0 && (17)\\
& x \text{ integer}, && (18)
\end{aligned}$$

where $X$ is a nonempty convex rational polyhedron and let $S$ be the set defined by the constraints (16)-(18) i.e.

$$S = \{x : x \in X,\ x \geq 0,\ x \text{ integer}\}.$$

There are two approaches for decomposing (14)-(18) in order to obtain an equivalent IP formulation. In the convexification approach, constraints (16)-(17) are replaced with the tighter constraint

$$x \in C = conv(S)$$

while keeping the same set of (integer) feasible solutions. Any $x \in C$ is written as the sum of a convex combination of (integer) extreme points of $C$ and a nonnegative linear combination of its (integer) extreme rays. The corresponding equation replaces constraints (16)-(17) and $x$ is substituted in (14) and (15). We obtain a new IP formulation for the original problem in terms of new variables $\lambda$, but integrality is required for original variables $x$. If integrality is required for variables $\lambda$, optimality and even feasibility may be lost (Ben Amor (1997)).

In the Discretization approach, based on a discretization theorem of Nemhauser and Wolsey (1988), any $x$ satisfying (16)-(18) is expressed as the sum of a binary convex combination of a finite set of integer points of $S$ and a nonnegative integer linear combination of (integer) extreme rays of $C = conv(S)$. Substituting in (14)-(18), we obtain an IP formulation of the original problem in terms of new variables $\lambda$ associated to the set of columns (points and rays). Integrality may equivalently be required for original variables $x$ or new variables $\lambda$ (Ben Amor (1997)).

**Remarks**

**1.** The set of rays can be chosen to be identical for both IP formulations that are obtained via decomposition, but the sets of points are generally different. However when the elements of $S$ have all their components in $\{0, 1\}$, both formulations have the same sets of columns.

**2.** Linear relaxations (LR) are obtained by dropping integrality requirements. First, since integrality requirements are partially taken into account inside the columns, both formulations produce stronger LP bound than the linear relaxation of the original formulation (14)-(17). However, when the set defined by (16)-(17) has the integrality property, the original and decomposed formulation have the same LP bound (Geoffrion (1974)).

Because the number of columns is very large and all needed columns may not be known in advance, linear relaxations are solved by column generation. Let $\pi$ be the dual solution at optimality of the restricted master at some CG iteration. For the convexification approach, the subproblem is

$$\begin{array}{rl} \min & (c - \pi^T A)x \\ \textit{subj. to} & x \in C. \end{array}$$

For the discretization approach, the subproblem is

$$\begin{array}{rl} \min & (c - \pi^T A)x \\ \textit{subj. to} & x \in S. \end{array}$$

In both cases, the attractive columns that are successively inserted in the master problem correspond to optimal solutions of the subproblem, being extreme points or extreme rays of $C$. Hence the two IP formulations have identical LP bounds.

**3.** For the convexification approach, integrality is required for the original variables $x$. Hence, branching schemes should be developed for these variables. This is somehow natural because such decisions are more likely to preserve the structure of the subproblem and to be more easily enforced in it. For the discretization approach, integrality may be required either for the original variables $x$ or the new variables $\lambda$. But, efficient branching schemes should be based on original variables for the same reasons or at least on the subproblem formulation. Indeed, for a branching scheme based on new variables $\lambda$, one must find a way to "express" decisions in terms of the original variables to be able to enforce them in the subproblem. Branching directly on new variables is untractable in practice due to the very large number of columns and especially, because forbidding the generation of any column by the subproblem may need obtaining the $k^{th}$ best solution of the subproblem at depth $k$ of the branch-and-bound tree.

**4.** Dantzig-Wolfe decomposition is well suited to problems with block-angular structure subproblems. Considering each block separately, giving raise to as many subproblems as many block, is generally preferred to considering only one subproblem. The main reasons are that the number of possible columns in the former case is much fewer than in the latter case, and that having a column for each block in the master problem allows implicitly taking into account many combinations of these columns whereas having one subproblem enforces the use of only one combination of these columns in the master problem (Ben Amor (2002)).

When all subproblems are identical, only one subproblem is solved at each column generation iteration and the master problem size can be significantly reduced by aggregating variables corresponding to different commodities (Vanderbeck (2000b), Ben Amor (2002)). We illustrate this issue in the next section while decomposing Kantorovitch formulation.

**5.** Enforcing decisions in the subproblem is a main issue when developing any branching scheme for new variables. In fact the structure of the subproblem may be highly affected by branching decisions and solution algorithms should be, at least, adapted. This is the case when branching decisions are made with respect to variables $\lambda$ and not all columns needed for an integer optimal solution are not extreme points or rays of the subproblem. Branching on original variables $x$ faces another kind of difficulties when many identical subproblems are considered. Aggregation of $\lambda$-variables breaks the symmetry in the master problem. It is of crucial importance to build a branching scheme that breaks, or at least highly reduces, symmetry between subproblems. In the binary case, i.e. all $\lambda$ variables take $0-1$ values, right-hand side of constraints are binary, and the subproblem is solved as a $0-1$ problem, this can be easily done. However, in the general integer case, this is a difficult task and to the best of our understanding, the choice of the original formulation and the corresponding subproblem is a key issue to the efficiency of branching schemes.

## 2.4 Decomposition of Kantorovitch formulation

Both convexification and discretization approaches can be applied to CSP giving raise to slightly different formulation. CSP has a block-diagonal structure and gives raise a $|K|$ subproblems. The set of integer feasible points of subproblem $k$ is

$$S_k = \{x = (x_0, \ldots, x_m)^T : \sum_{i=1}^{m} x_i w_i \leq W,\ x \geq 0,\ x \text{ integer}, x_0 \in \{0,1\}\}.$$

First, all set $S_k$, and hence all subproblems, are identical ($S_k = S, \forall k = 1, \ldots, K$). Note that $S$ is bounded. If the binary component $x_0$ of $x$ takes value 0, all other components equal 0; this is the empty pattern. When it takes value 1, any feasible pattern (including the empty one) may be represented by the values of the other components. Because the number of used rolls is minimized, the empty pattern will never part of the solution with component $x_0 = 1$. Let $P_0$ and $P$ denote the set of all feasible patterns and the set of nonempty patterns, respectively. In a same manner we define $\Omega_0$ and $\Omega$ the set of all feasible patterns and the set of nonempty patterns that are extreme points of $C = conv(S)$. The index 0 is associated to the empty pattern in $P_0$ and $\Omega_0$.

$\{x^p\}_{p\in\Omega_0}$ being the set of extreme points of $C$, any point of $x^k$ in $C_k$ can be expressed as a convex combination of these points.

$$x^k = \textstyle\sum_{p\in\Omega_0} \lambda_p^k x^p, \quad \displaystyle\sum_{p\in\Omega_0} \lambda_p^k = 1, \lambda_p^k \geq 0, \forall p \in \Omega_0.$$

Substituting in (1)-(5), we obtain the formulation (Ben Amor (1997), Vance (1998))

$$\min \quad \sum_{p\in\Omega} \sum_{k=1}^{K} \lambda_p^k \tag{19}$$

$$\text{subj. to} \quad \sum_{p\in\Omega}\sum_{k=1}^{K} x_{pi}\lambda_p^k \geq b_i \tag{20}$$

$$\sum_{p\in\Omega_0} \lambda_p^k = 1, k = 1,\ldots,K \tag{21}$$

$$\lambda_p^k \geq 0, \forall p \in \Omega_0 \tag{22}$$

$$x^k = \sum_{p\in\Omega} \lambda_p^k x^p, x^k \text{ integer}, k = 1\ldots,K. \tag{23}$$

A slightly different formulation can be obtained by considering the set of all points of $S$. Any $x^k \in S_k$ is written as a binary convex combination of all points in $S$.

$$x^k = \ \textstyle\sum_{p\in\Omega_0} \lambda_p^k x^p, \quad \displaystyle\sum_{p\in P_0} \lambda_p^k = 1, \lambda_p^k \in \{0,1\}, \forall p \in P_0.$$

Substituting in (1)-(5), we obtain the formulation

$$\min \quad \sum_{p\in P}\sum_{k=1}^{K} \lambda_p^k \tag{24}$$

$$\text{subj. to} \quad \sum_{p\in P}\sum_{k=1}^{K} x_{pi}\lambda_p^k \geq b_i \tag{25}$$

$$\sum_{p\in P_0} \lambda_p^k = 1, k = 1,\ldots,K \tag{26}$$

$$\lambda_p^k \geq 0, \forall p \in P_0,\ k = 1\ldots,K \tag{27}$$

$$\lambda_p^k \in \{0,1\} \forall p \in P_0.\, k = 1\ldots,K \tag{28}$$

where integrality may equivalently be required for variables $x^k$.

The two formulations present two differences: the sets of columns are not the same. Formulation (24)-(28) has additional columns that are not extreme points of the knapsack polytope and integrality can be required for variables $\lambda^k$. However, following Remark 2 of the precedent section, they have the same LP bound and the same column generation subproblem. Aggregation is advised for both formulations. We present it in a general manner that may be applied to any other IP problem. Define the integer variable $\lambda_p$ as

$$\lambda_p = \sum_{k=1}^{K} \lambda_p^k, \forall p. \tag{29}$$

Such variable counts in fact the number of rolls cut owing to pattern $p$ for $p \in \Omega$, and the number of unused patterns for $p = 0$. Replacing in either formulation, index $k$ disappears from the objective function and covering constraints. Computing a $\lambda_p^k-$solution from a $\lambda-$solution is always possible. We have in fact a transportation problem ($|\Omega_0|$ supply ($= \lambda_p$) nodes and $K$ demand ($= 1$) nodes whose feasibility conditions is $\sum_{p\in\Omega_0} \lambda_p = K$ which follows trivially from (29). This argument holds for both continuous and integer cases. In particular, in the discretization approach, integrality may be required for variables $\lambda_p$ leading to the formulation of Gilmore and Gomory (8)-(10). In the convexification approach, the obtained formulation

$$\min \quad \sum_{p\in\Omega} \lambda_p \tag{30}$$

$$\textit{subj. to} \quad \sum_{p\in\Omega} x_{pi}\lambda_p \geq b_i \tag{31}$$

$$\sum_{p\in\Omega_0} \lambda_p = K \tag{32}$$

$$\lambda_p \geq 0, \forall p \in \Omega_0 \tag{33}$$

$$\lambda_p = \sum_{k=1}^{K} \lambda_p^k, p \in \Omega_0 \tag{34}$$

$$\lambda_p^k \geq 0, \forall p \in \Omega_0 \tag{35}$$

$$x^k = \sum_{p\in\Omega} \lambda_p^k x^p, x \text{ integer}, k = 1, \ldots, K. \tag{36}$$

is basically different from the one of Gilmore and Gomory. It considers a subset of the set of columns used in (8)-(10) and integrality is still required for variables $x^k$.

Linear relaxations obtained by dropping integrality constraints are solved by column generation. If $\bar{\pi}$ is the dual optimal solution of the restricted master problem at some column generation iteration, the reduced cost of any column $x^p$ is $1 - \sum_{i=1}^m \bar{\pi}_i x_{pi}$. Since generating the most attractive column aims at finding the minimum reduced cost column, that is an integer feasible knapsack solution, the subproblem is formulated as

$$\begin{aligned} \textit{max} \quad & \sum_{i=1}^m \bar{\pi}_i x_i \\ & \sum_{i=1}^m x_i w_i \leq W \\ & x_i \geq 0 \text{ and integer} \end{aligned}$$

which is the same as the one of G-G formulation (11)-(13).

It is important to note that since the subproblem objective function is linear, only feasible points that are extreme points (or at most on the boundary) of $C = \mathit{conv}(S)$ can be generated by solving the knapsack problem. The corresponding columns are enough to produce an integer solution of CSP if branching decisions are taken on $x^k-$variables (convexification approach) while keeping the same subproblem whereas they are not sufficient if decisions are taken directly on $\lambda-$variables (discretization approach) without altering the subproblem structure. This is illustrated by the following example (Ben Amor (1997)). Let the CSP having roll length $W = 6$, two items of lengthes $w_1 = 2$ and $w_2 = 3$ with corresponding demands $b_1 = 4$ and $b_2 = 3$. The feasible patterns that may be generated are $(3, 0)$ and $(0, 2)$. The empty pattern $(0, 0)$ cannot be generated by the subproblem and may be trivially taken into account.The optimal solution consists of using patterns $(3, 0)$, $(0, 2)$ and $(1, 1)$ exactly once. But the last pattern is an interior point to the set of feasible patterns and will never be generated without modifying the subproblem. However it can be expressed as a convex combination of extreme pattern as follows: $(1, 1) = (1/6)(0, 0) + (1/3)(3, 0) + (1/2)(0, 2)$. However symmetry is a critical issue when dealing with branching schemes based on $x^k-$variables. The ways of obtaining an integer solution to CSP are addressed later in the paper.

## 2.5 Arc-flow model

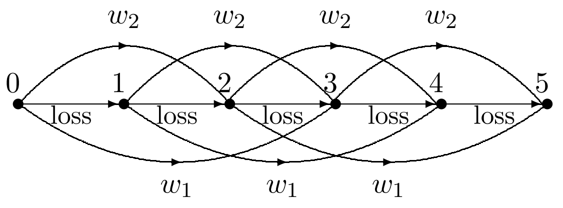
Valério de Carvalho (1999) proposed an arc-flow model for the integer solution of BPP. Given bins of integer capacity $W$ and a set of different item sizes $w_1, w_2, \dots, w_m$, the problem of determining a valid solution to a single bin can be modelled as the problem of finding a path in an acyclic directed graph, $G = (V, A)$, with $V = \{0, 1, 2, \dots, W\}$ and $A = \{(i, j) : 0 \le i < j \le W \text{ and } j - i = w_d \text{ for every } d \le m\}$, meaning that there exists a directed arc between two vertices if there is an item of the corresponding size. Consider additional arcs between $(k, k+1), k = 0, \dots, W-1$, corresponding to unoccupied portions of the bin. There is a packing in a single bin iff there is a path between vertices 0 and W. The lengths of the arcs in the path define the item sizes to be packed.

**Example 1** Figure 1 shows the graph associated with an instance with bins of capacity $W = 5$ and items of sizes 3 and 2. In the same Figure, a path is shown that corresponds to 2 items of size 2 and 1 unit of loss. □

Shapiro (1968) used this kind of formulation to model the knapsack problem as the problem of determining the longest path in a directed graph. Likewise, it can be used to model BPP. If a solution to a single bin corresponds to the flow of one unit between vertices 0 and W, a path carrying a larger flow will correspond to using the same packing solution in multiple bins.

By the flow decomposition properties (see Ahuja et al. (1993)), non-negative flows can be represented by paths and cycles. The graph $G$ is acyclic, and any flow can be decomposed in directed paths connecting vertex 0 to vertex W. A solution with integer values of flow in every arc, can be transformed into an integer solution to the BPP.

The problem is formulated as the problem of determining the minimum flow between vertex 0 and vertex W with additional constraints enforcing that the sum of the flows in the arcs of each order must be greater or equal to the number of items of a given size. Decision variables $x_{ij}$, associated with the arcs defined above, correspond to the number



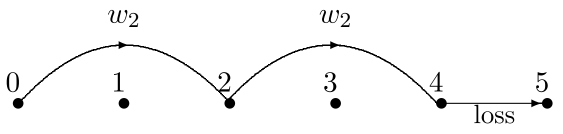


Figure 1: Graph and a cutting pattern

of items of size $j - i$ placed in any bin at the distance of $i$ units from the beginning of the bin. The number of variables is $O(mW)$. The model is as follows:

$$\min z \tag{37}$$

subject to

$$+\sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = \begin{cases} -z & \text{, if } j = 0 \\ 0 & \text{, if } j = 1,\dots,W-1 \\ z & \text{, if } j = W \end{cases} \tag{38}$$

$$\sum_{(k,k+w_d)\in A} x_{k,k+w_d} \geq b_d \ , \ d = 1,2,\dots,m \tag{39}$$

$$x_{ij} \geq 0 \ , \ \forall (i,j) \in A \tag{40}$$

$$x_{ij} \text{ integer } \ , \ \forall (i,j) \in A \tag{41}$$

If we apply Dantzig-Wolfe decomposition to (37)–(40) keeping (37) and (39) in the master problem, the subproblem defined by (38) and (40) is a flow problem with a solution space that correspond to the valid flows between vertex 0 to vertex W (Ben Amor (1997), Valério de Carvalho (1999)).

Actually, the variable $z$ can be seen as a feedback arc from vertex W to vertex 0 (could also be denoted as $x_{W0}$) and the solutions to the subproblem as circulation flows, which include a path between vertices 0 and W and the arc $x_{W0}$. There is a one-to-one correspondence between circulations and paths. If we see the subproblem solutions as circulations, the subproblem constraints define a homogeneous system, and is unbounded. Therefore, the corresponding polyhedron has a single extreme point, the null solution, and a finite set of extreme rays, which are the directed circulations, each corresponding to a valid pattern. The subproblem will only generate extreme rays, and the substitution of the patterns in (37) and (39) results in the Gilmore-Gomory model (8)-(10), which is a

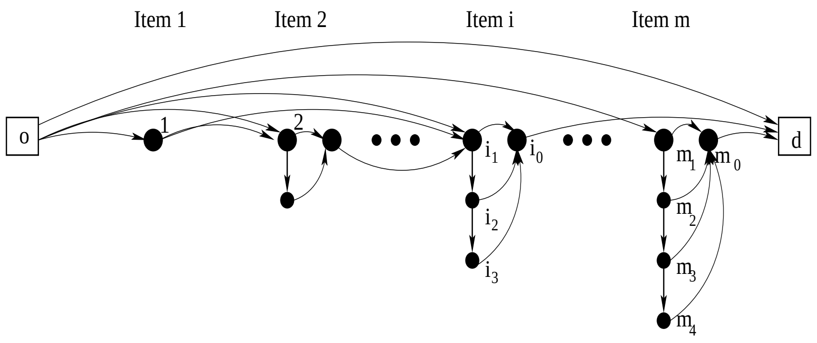


Figure 2: An acyclic VRP network for CSP.

nonnegative linear combination of the patterns, with no convex combination constraint. As the two models are equivalent, the lower bounds given by their LP relaxations are equal.

The subproblem has the integrality property and hence both original and decomposed formulations have the same LP bound. Moreover, all feasible patterns may be described by a path from 0 to $W$ and vice versa. Hence either convexification or discretization approaches results produces Gilmore and Gomory formulation.

## 2.6 An acyclic capacitated VRP model

Ben Amor (1997) consider feasible patterns as routes in an acyclic capacitated VRP network (see figure 2 for an illustration). First since the order in which items are cut on a roll has no effect on the cost function, items are ordered in non-increasing order of length, i.e. $w_i \leq w_{i+1}, i = 1, \ldots, m-1$. Each roll correspond to a vehicle of capacity $W$. A pair of fictive origin and destination depots represent the start $(o(k))$and end $(d(k))$ nodes of a route (pattern) of vehicle $k$ $(k = 1, \ldots, K)$. To item (client) $i$ corresponds a set of $n_i = \lfloor W/w_i \rfloor$ nodes $i_1, \ldots, i_{n_i}$ and a supplementary node $i_0$. Two types of arcs are used within this set of nodes: $(i_v, i_{v+1})$ $(v = 1, \ldots, n_i - 1)$ and $(i_v, i_0)$ $(v = 1, \ldots, n_i)$. Any route (pattern) visiting item $i$, begins with node $i_1$ and leaves at node $i_0$. An inter-task arc joining item $i$ and item $j$ $(j > i)$ exists if $w_i + w_j \leq W$. Such an arc starts at node $i_0$ and ends at nodes $j_1$. The first, respectively the last, arc of a route takes the form $(o(k), i_1)$ $(i = 1, \ldots, m)$, respectively $(i_0, d(k))$ $(i = 1, \ldots, m)$. An additional arc $(o(k), d(k))$ corresponds to the empty pattern. To represent a feasible pattern, a route $R$ has to respect the knapsack constraint $\sum_{i \in R} w_i \leq W$. A resource is used to compute the load of a vehicle (roll) at each node of the network. Arcs of types $(o(k), i_1)$ $(i = 1, \ldots, m)$, $(j_0, i_1)$, and $(i_v, i_{v+1})$ $(v = 1, \ldots, n_i - 1)$ have resource consumption $w_i$. Other arcs have 0 resource consumption. An upper limit of $W$ on the resource amount used up to any node, except the origin $o(k)$, is imposed. Let $N$ the set of task nodes, i.e. $N = \cup_{i=1}^{m} \{i_v, v = 0, \ldots, n_i\}$, and $A_k$ the set of arcs corresponding to vehicle (roll) $k$. To each arc $(i, j) \in A_k$ we associate a binary variables $x_{ij}^k$. CSP is then formulated as a multicommodity flow problem where the number of nonempty routes is minimized.

$$min \quad \sum_{k \in K} \sum_{j=1}^{m} x^k_{o(k)j_1} \tag{42}$$

$$subject\ to$$

$$\sum_{k \in K} (\sum_{v=1}^{n_i} \sum_{(j,i_v) \in A_k} x^k_{ji_v}) \geq b_i, i = 1, \dots, m \tag{43}$$

$$\sum_{(o(k),j) \in A^k} x^k_{o(k)j} = 1, k \in K \tag{44}$$

$$\sum_{(i,j) \in A^k} x^k_{ij} - \sum_{(j,i) \in A^k} x^k_{ji} = 0, i \in N, k \in K \tag{45}$$

$$\sum_{(i,d(k)) \in A^k} x^k_{id(k)} = 1, k \in K \tag{46}$$

$$\sum_{i=1}^{m} w_i (\sum_{v=1}^{n_i} \sum_{(j,i_v) \in A^k} x^k_{ji_v}) \leq W(\sum_{j=1}^{m} x^k_{o(k)j_1}), k \in K \tag{47}$$

$$x^k_{ij} \in \{0, 1\}, k \in K, (i,j) \in A^k. \tag{48}$$

The objective function (42) aims at minimizing the number of nonempty routes (patterns). Each constraint (43) requires that the number of items $i$ cut on all rolls satisfy the demand $b_i$ and constraints (44)-(46) enforce flow conservation for each commodity $k$ while requiring that 1 unit of flow be shipped from $o(k)$ to $d(k)$. Finally, (linear) constraints (47) ensure that capacity limit for any used rolls is not overpassed and constraints (47) require that all variables be binary.

Applying either the convexification or the discretization approach of extended D-W decomposition to this formulation leads to Gilmore and Gomory model, with a capacitated shortest path as subproblem. Integrality may equivalently be required either for $\lambda-$variables or $x^k-$variables. It is more natural and easier to develop a branching scheme based on $x^k-$variables. However, there is a major issue here, symmetry between rolls, that must be taken in account. Branching schemes developed for several models are discussed later in the paper.

Formulation (42)-(48) may be seen as a disaggregated form of Kantorovitch formulation (1)-(4). However, there is a difference that all patterns are extreme points of the subproblem (44)-(48) and hence can be generated, if needed, by modifying arc costs. This is indeed done by considering different dual variable values during column generation. It is sufficient to modify the arc costs in an adequate manner, e.g. generating different dual variables values at some column generation iteration (Ben Amor (1997)).

The binary case, i.e. all demands $b_i$ equal 1 and any item may not be cut more than once on a pattern, present some interesting particularities. First, each item is represented by a single node and must be visited exactly once by a single root. This allows the aggregation of all commodities and index $k$ is no longer needed in the formulation. This has an important effect on the branching strategy as will be seen later.

## 3 Integer solutions

Up to the nineties, it was recognized that it was not easy to combine column generation with tools to obtain integer solutions to the CSP. We quote the final comments in Gilmore (1979):

> "A linear programming formulation of a cutting stock problem results in a matrix with many columns. A linear programming formulation of an integer programming problem results in a matrix with many rows. Clearly a linear programming formulation of an integer cutting stock problem results in a matrix that has many columns and many rows. It is not surprising that it is difficult to find exact solutions to the integer cutting stock problem."

### 3.1 Heuristics

Gilmore and Gomory (1961) suggest rounding to obtain good quality integer solutions. Rounding up the fractional variables of the optimal solution of the column generation model guarantees a heuristic solution of value $z_H : z_H \leq z_{LP} + m$, where $z_{LP}$ is the optimum of the LP relaxation. The First Fit Decreasing (FFD) heuristic provides solutions for the BPP, and the CSP, with an absolute performance ratio of 3/2, i.e., $z_H \leq 3/2\ z^*$, where $z^*$ is the value of the optimum (Simchi-Levi (1994)). This ratio can be improved to 4/3, and this bound is tight, if a heuristic based on the solution of Gilmore-Gomory model is used (Chan et al. (1998)).

Rather than simply rounding up, more elaborate rounding heuristics were devised to improve effectiveness. Wäscher and Gau (1996) did extensive computational experiments with instances with average demands of 10 and 50, and found the optimal solutions in almost all cases. However, if the average demand is very low, as in the BPP, where it can be close to one or even equal to one, the variables are often a fraction of unity, and it is not so easy for heuristics to find the optimum. Other combinatorial enumeration techniques can be used in this case, as MTP (Martello and Toth (1990)) or BISON (Scholl et al. (1997)). The value of the lower bound of Gilmore-Gomory solution has also been used to prove the optimality of solutions obtained with MTP (Schwerin and Wäscher (1999)).

To generate the initial solution for column generation, instead of using FFD, some authors resort to pseudo-polynomial heuristics, based on the solution of a series of knapsack problems. These are greedy procedures that iteratively select the best cutting pattern using the items in a list. Initially, the list has all the items; at each iteration, the items in the knapsack solution are removed, and the process is repeated until the list is exhausted. Their computation time is not significant in the framework used, and they generally provide better starting solutions, with, at least, some very good cutting patterns, even though the last patterns may be very poor (see, for instance, Vanderbeck (2000b); Valério de Carvalho (2003) for a definition of the knapsack problems used).

## 3.2 Branch-and-price

In the nineties, several attempts to combine column generation with branch-and-bound succeed in obtaining the optimum integer solution of larger instances of some integer programming and combinatorial optimization problems. This technique has been denoted as branch-and-price. Stronger LP models and good quality lower bounds are of vital importance when using LP based approaches to solve integer problems. Branch-and-price also proved to be a useful framework for the solution of quite large instances of both CSP (1/V/I/R) and BPP (1/V/I/M), that other combinatorial enumeration branch-and-bound algorithms failed to solve to optimality.

In branch-and-price, it is desirable to ensure compatibility between the restricted master problem and the subproblem. First, the partition rule should not induce intractable changes in the structure of the subproblem. Desirably, it should remain the same optimization problem both during the solution of the LP relaxation and the branch-and-price phase. The second issue is symmetry. Branching strategies should be devised that partition the solution space in mutual exclusive sets, which are explored in different nodes of the branch-and-bound search tree. Symmetry is detrimental if the same solution is explored in different nodes of the branch-and-price tree. Other issues are also important to obtain more robust algorithms. Balanced partition rules should be selected: the branching constraints should partition the solution set evenly among subtrees. It is also desirable to select the branching constraints so that stronger decisions are taken at lower levels of branch-and-bound tree. Most applications of branch-and-price are for problems with binary variables (for a review, see Barnhart et al. (1998)). Finally, note that CSP have general integer variables, not restricted to be binary.

The strategy of imposing branching constraints directly on the variables of the reformulated model poses the following difficulty: a column that is restricted by a branching constraint in the master problem may turn out to be most attractive column generated by the subproblem. To deal with this problem, some authors keep track of the columns already present in the master problem that must not be regenerated. The subproblem has to be solved by an enumeration scheme that rejects the forbidden columns (Degraeve and Schrage (1999); Degraeve and Peeters (2003); Belov (2003)). It is widely accepted that, for most integer programming problems, the best strategy to combine column generation with branch-and-bound is to use branching constraints based on the variables of the original model (Desrosiers et al. (1995)).

## 3.3 Branching schemes

Branching strategies are related to the original formulation that is decomposed to produce the IP mater problem, or at least to the kind of subproblem used to generate columns. For CSP, there three possible original formulation: Kantorovitch formulation (1)-(4), the arc-flow formulation (37)-(41), and the VRP formulation (42)-(48).

## Kantorovitch formulation

Even if it appears to be more natural, efficiency of branching on variables $x^k$ is compromised due to the symmetry between rolls. For instance, fixing any $x_0^k$ to 1 or 0, meaning that roll $k$ is used or not, has no effect on the problem because all rolls are identical. Once the number of rolls (or a lower and/or upper bound) at optimality is known, one may fix $x_0^k$ variables corresponding to these rolls to 1 and others to 0, so that many variables are eliminated from the problem. Moreover, bounding any $x_i^k$ ($k = 1, \ldots, K,\ i = 1, \ldots, m$) by any integer value does not eliminate any fractional solution because such a solution may be retrieved by using another roll $k$ for item $i$.

The solution of CSP is composed of patterns and each pattern is composed of a set of items to be cut together on a same roll. Hence at items level, the information that is useful to build a solution should give either a set of items to be/not to be cut on a same set of rolls, or a maximum or/and minimum numbers of copies of an item to be cut on a set of rolls.

**Vanderbeck's rule** Vanderbeck compared several branching schemes and suggested the use of one that is based on the binary representation of the cutting pattern columns of the Gilmore-Gomory model. The binary representation of a column $A^p$ is a 0-1 vector $A^{p'}$ with size $m' = \sum_{d=1}^{m} m_d$, with $m_d = \lceil log_2(l_d^{max}+1)\rceil$, being $l_d^{max}$ the upper bound on the number of copies of item $d$ in a cutting pattern, as defined above. We will denote the elements of $A^{p'}$ as $a_k^{'}, k = 1, \ldots, m'$, dropping the index $p$, for the sake of clarity. The binary representation is such that the number of items produced for order $d$ is $a_d = \sum_{j=0}^{m_d-1} 2^j a_{p_d+j}^{'}, d = 1, \ldots, m$, where $p_d = 1 + \sum_{j=1}^{d-1} m_j$.

Given a fractional solution of Gilmore-Gomory model, it is always possible to find subsets of rows $O$ and $P \subset \{1, \ldots, m'\}$, and a subset of columns

$$\hat{P} = \{p \in P : a_k^{'} = 0, \forall k \in O \text{ and } a_k^{'} = 1, \forall k \in P\}$$

such that $\alpha = \sum_{p\in\hat{P}} \lambda^p$ is fractional. The branching constraints are: $\sum_{p\in\hat{P}} \lambda^p \geq \lceil\alpha\rceil$ and $\sum_{p\in\hat{P}} \lambda^p \leq \lfloor\alpha\rfloor$. If it possible to identify sets, such that $|O| + |P| = 1$, the branching rule leads to very easy modifications in the subproblem, both in the left and in the right branches. This branching scheme has been used along with a combination of heuristics (the best of BFD, FFD, and a pseudopolynomial heuristic he proposed) and Martello and Toth (1990) lower bounds. Experiments with CSP instances, where the values of the demands are large, show that the procedure is quite robust and powerful. However, when demands are very small, as in the BPP, it may be necessary to select sets with $|O|+|P| \geq 2$, leading to a subproblem that is no longer a knapsack problem, but an extended knapsack problem with new extra binary variables, needed to identify the attractive columns correctly. Computational experiments show results comparable to the ones obtained with the arc-flow model for the instances under study (Vanderbeck (1999, 2000b)).

**Vance rule** Vance (1998) first proposed ed a branching scheme based on integrality requirements in (19)-(23). This strategy suffers however from the symmetry between rolls, as said before, and was beaten by a quiet complex branching strategy based on Gilmore and Gomory model variables for medium size problems. This latter strategy is based on the total flow of columns containing a certain minimum number of copies of items from a set $S$. It suffers two major drawbacks: there's no guaranty on the size of $S$ and the subproblem is no longer a knapsack since many additional variables are needed. Owing to the author, it should become untractable even for small depth nodes of the branching tree. An improving strategy is based o the use of maximal patterns, but it amounts to branching directly on $\lambda_p$ variables. The structure of the subproblem can be preserved for one branch thanks to a specialized algorithm and is highly affected for the other branch (even the maximal pattern property is lost). This last strategy has been successfully used to solve medium size problems.

## Arc-flow model

The arc-flow model provides a branching scheme for a branch-and-price algorithm for the CSP that preserves the structure of the subproblem, which is as a longest path problem in an acyclic digraph with modified costs, that can be solved using dynamic programming. Valério de Carvalho (1999) implemented a column generation algorithm based on a master problem with variables of the arc-flow model. It starts with a subset of arcs corresponding to an initial solution, and new arcs are added to the master problem if they belong to an attractive path and are not already present in the master problem. In the arc-flow model, there are flow conservation constraints for the nodes. If the new arcs are incident into nodes not previously considered, new constraints are explicitly added to the formulation. Branching constraints are imposed on single arc-flow variables of the master problem of the following type, where $\alpha = x_{ij}$ is fractional:

$$x_{ij} \leq \lfloor \alpha \rfloor \tag{49}$$

and

$$x_{ij} \geq \lceil \alpha \rceil \tag{50}$$

The model has symmetry, because different paths may correspond to the same cutting pattern. In instances with a small average number of items per bin, which happen to be rather difficult instances, if reduction criteria are used, there is low symmetry, and its undesirable effects are not so harmful. Computational results show the optimal solutions of all the bin packing instances of the OR-Library (Beasley (1990)). These instances have demands of few items of each size. For example, the larger instances, of the $t501$ class, have about 200 different item sizes and a total of 501 items, yielding an average demand of about 2.5.

A different strategy that eliminates symmetry and preserves the structure of the subproblem is to use a master problem with the columns of the reformulated model of Gilmore-Gomory and branching constraints based on the arc-flow variables, which are explicitly added to the formulation (Ben Amor (1997), Alves and Valério de Carvalho (2003)). Any column of Gilmore-Gomory's model involves a unique set of arc-flow variables, if we consider that items are placed by decreasing value of width. Branching constraints imposed on a given arc-flow will constrain the value of a definite set of columns of the reformulated model. Penalties and prizes resulting from branching constraints of the type greater-than-or-equal-to or less-than-or-equal-to, respectively, only affect the reduced cost of the corresponding arc in the subproblem.

Let $\pi_d, d = 1, \ldots, m$, be the dual variables associated with the demand constraints, and $\mu$ and $\nu$ the vectors of dual variables associated with the branching constraints of each type Let $G^w_{(i,j)} \subseteq G^w$ and $H^w_{(i,j)} \subseteq H^w$ be the sets of branching constraints imposed on the specific arc $(i,j)$ at a given node $w$ of the branch-and-bound tree. The reduced cost of variable $x_{ij}$ at node $w$ is $\bar{c}_{ij} = \pi_d - \sum_{l \in G^w_{(i,j)}} \mu_l + \sum_{l \in H^w_{(i,j)}} \nu_l$, where $d$ is the item that corresponds to arc $(i,j)$. Only the costs change, and the subproblem structure remains unchanged. It may be solved using dynamic programming as a knapsack problem that only selects cutting patterns with items placed by non-increasing width. The recursive equations are as follows:

$$\begin{aligned} F_0(0) &= 0 \\ F_d(x) &= \max_{valid\ l:\ 0 \le l \le l_d^{max}} \{F_{d-1}(x - lw_d) + \sum_{k=0}^{l-1} \bar{c}_{x-(k+1)w_d, x-kw_d}\}, \\ & \quad x = 0, 1, \ldots, W, \quad d = 1, ..., m \end{aligned}$$

where $l_d^{max} = min(b_d, \lfloor W/w_d \rfloor)$ is the upper bound on the number of copies of item $d$ in a cutting pattern, which is limited by the demand for item $d$ and the size of the roll.

This branching scheme can be easily extended to branching constraints based on sets of arcs incident on a given node. In this case, the dual variable of a branching constraint acts on all arcs in the set. Nevertheless, the computational burden is heavier for instances with larger values of roll widths.

## VRP model

Integrality may be required equivalently either for variables $x^k_{ij}$, $\lambda^k_p$, or $\lambda_p$. In the first two cases symmetry is very harmful to branching scheme efficiency. Aggregating flows on arcs, one obtains variables

$$x_{ij} = \sum_{k=1}^{K} x^k_{ij}$$

that count the number of columns (rolls) using arc $(i,j)$. These variables must be integer for any integer solution to CSP. Hence, for any solution such that $x_{ij} = \alpha$ is fractional for some arc $(i,j)$ one creates two nodes by adding either the constraint

$$x_{ij} \le \lfloor \alpha \rfloor$$

or the constraint

$$x_{ij} \geq \lfloor \alpha \rfloor$$

to the master problem. The dual variables associated to these constraint will affect the arc costs in the subproblem at each column generation iteration and the subproblem structure remains unchanged. Since all columns (patterns) are extreme points of the subproblem, any column that is needed to attain integer optimality can be generated in this way.

However in the general integer case, i.e. a node may be visited by more than one path in an optimal integer solution, one can obtain a fractional solution ($\lambda_p$ fractional) while all $x_{ij}$ are integer (an example is given in Ben Amor (1997)). In this case, there exists at least one item $i$ or which an associated node is visited more than once in the solution. A new item with demand 1 is created while the demand of $i$ is decreased by 1. The master problem covering constraints are modified following these changes. The worst case happens when all items are completely disaggregated and the problem becomes a BCSP where all items have demand = 1. In this case the branching scheme using aggregated flow variables is convergent since

$$x_{ij} \in \{0, 1\}, \forall (i, j) \Leftrightarrow \lambda_p \in \{0, 1\}, \forall p.$$

This branching scheme preserves the structure of the subproblem as a constrained shortest path and decisions are directly enforced in the subproblem while the master problem size remains unchanged. Constraints of the type $x_{ij} = 0$ simply amount to removing the arc from the subproblem. On the other hand, the constraints of the type $x_{ij} = 1$ amount to removing all arcs out of node $i$ and all arcs into node $j$. It was successfully used to solve the VRP with time windows using branch-and-price (Desrochers et al. (1992)) and proved to be efficient for BCSP (Ben Amor (1997)). The main difficulty in BCSP is the solution of linear relaxation by column generation. But Ben Amor (2002) showed that this may be done as efficiently as for the corresponding CSP either by aggregating identical size items or by using deep dual-optimal inequalities (Ben Amor et al. (2003)).

Branching on aggregated flow variables has many interesting properties. First, it allows the use of several score functions to choose the branching variable. Scores take into account the weights of items. Lower bounds and preprocessing procedure of Martello and Toth (1990) can be extended to the problems obtained at each branch-and-bound node. The experiments carried out by Ben Amor (1997) show that no disaggregation has been necessary and the branching scheme turns out to be efficient for usually used test problems. Even for the binary case, branching on aggregated flow variables turns out to be very efficient. Also fixing several variables at once proved to be efficient in a depth first strategy and no backtracking has shown to be necessary. This due to the fact that nearly all solved problems have the integrality property, i.e. the optimal integer value is obtained by rounding up the linear relaxation optimal value.

## General comments

An efficient branching scheme for column generation should have a smaller tree than the one resulting from branching directly on $\lambda-$variables. The rules of Vanderbeck and Vance have the drawback of significantly modifying the subproblem structure. Moreover, the number of possible branching nodes is very high. Branching scheme based on VRP model has the advantage of preserving a constrained shortest path as subproblem. Possible disaggregation may lead to subproblem of larger size deeper in the branching tree which breaks the rule that deeper in the branching tree, subproblems and desirably master problems should become easier to solve. Compared to this method, the branching based on the arc-flow model presents the advantage that no disaggregation is needed. Moreover, both strategies allow using more sophisticated branching rules based on $x_{ij}$ variables. Besides all these intrinsic differences, all branching schemes lead to efficient solution of classical test problems. This is due to the fact that these problems have zero gap, the use of depth first strategy, and the use of several heuristics.

## 4 Stabilization

Column generation processes are known to have a slow convergence and degeneracy problems. There are often large oscillations in the values of the dual variables from one iteration to the next. Primal degeneracy also arises: in many iterations, adding new columns to the restricted master problem does not help to improve the objective value. Recent computational experiments show that these problems can be mitigated using *dual-optimal inequalities* (Ben Amor et al. (2003)) and *stabilization methods* (Ben Amor (2002)).

From the dual standpoint, column-generation processes can be viewed as dual cutting-plane algorithms (Kelley Jr. (1961)), in which the restricted set of variables used to initialize the restricted master problem provides a first relaxation of the dual space. Clearly, better heuristics for the starting solution provide tighter relaxations. Then, at each iteration, dual feasibility cuts are added to the model to eliminate the previous undesired dual solution. The dual-space relaxation is successively tightened, until a feasible dual solution to the entire problem is found.

Consider the LP relaxation of the CSP, $\min\{cx : Ax = b, x \geq 0\}$, where the columns of $A$ correspond to valid cutting patterns, whose dual is $\max\{\pi b : \pi A \leq c\}$. The following inequalities are a family of *dual-optimal inequalities* (Ben Amor (2002); Valério de Carvalho (2003)), meaning that they are valid inequalities for the optimal dual space of the CSP. Any optimal dual solution will obey:

$$-\pi_i + \sum_{s \in S} \pi_s \quad \leq \quad 0, \ \forall i, S, \tag{51}$$

for any given width $w_i$, and a corresponding set $S$ of item widths, indexed by $s$, such that $\sum_{s \in S} w_s \leq w_i$. Intuitively, from the primal point of view, these columns mean that an item of a given size $w_i$ can be split, and used to fulfill the demand of smaller orders, provided the sum of their widths is smaller than or equal to the initial size.

If we add at initialization time a set of dual-optimal inequalities to the dual problem, $\pi D \leq d$, we get the following primal-dual pair:

$$\begin{array}{lll|lll} \min & cx + dy & & \max & \pi b \\ s.t. & Ax + Dy = b & & s.t. & \pi A \leq c \\ & x, y \geq 0 & & & \pi D \leq d \end{array}$$

The motivation for the use of dual-optimal inequalities is the following: the dual space is restricted during all the column generation iterations, but the new columns added in the primal problem, relaxing the primal space, are not a problem, because eventually it is possible to recover an optimal solution to the original problem. Valério de Carvalho (2003) added a set of dual-optimal inequalities from sets $S$ of small cardinality ($|S| \leq 2$) : from sets of cardinality 1, $-\pi_i + \pi_{i+1} \leq 0, i = 1, \ldots, m-1$, and from sets of cardinality 2, inequalities of the type $-\pi_i + \pi_j + \pi_k \leq 0$, one for each value of $i$, generated using the smallest index $j$ (corresponds to largest width $w_j$), if such value existed, for which there was a $k$ such that $w_i \geq w_j + w_k$. The total number of dual-optimal inequalities is less than $2m$.

Computational experiments show a sensible reduction in the number of columns generated and degenerate iterations. The savings are more impressive in larger, more difficult instances, when there is an explosion in the number of possible columns. For some instances, the speed-up factor is approximately 4.5, and the percentage of degenerate iterations falls from approximately 39.8% to about 8.5%. Ben Amor (2002) conducted a similar study on the classical test problems and another set of more difficult test problems. Results show an impressive reduction in the master problem cpu time and especially the number of column generation iterations.

The dual-optimal inequalities correspond to cycles in the space of the arc-flow variables, where exactly one arc (the one corresponding to item $i$ in Equation (51)) is traversed in the direction opposite to its orientation. Combining a path and a cycle produces a new valid path. Valério de Carvalho (2003) provides an algorithm to drive the $y$ variables to 0, when they take a positive value in the optimal solution, retrieving a valid optimal primal solution to the original CSP. Actually, this is not needed, if a perturbation technique is used, that amounts to giving a $\varepsilon$ cost to the cycles (Ben Amor et al. (2003)).

Even better results can be obtained if we use inequalities, that will be denoted as *deep dual-optimal inequalities*, that cut portions of the dual optimal space, but preserve, at least, one dual optimal solution (Ben Amor et al. (2003)). If a dual-optimal solution for the problem, $\tilde{\pi}^*$, is known in advance, the following stabilized primal and dual problems can be used:

$$\begin{array}{ll|ll} \min & cx - (\tilde{\pi}^* - \Delta)y_1 + (\tilde{\pi}^* + \Delta)y_2 & \max & \pi b \\ s.t. & Ax - y_1 + y_2 = b & s.t. & \pi A \leq c \\ & x \geq 0,\ y_1 \geq 0,\ y_2 \geq 0 & & \tilde{\pi}^* - \Delta \leq \pi \leq \tilde{\pi}^* + \Delta. \end{array}$$

where $\Delta > \mathbf{0} \in \mathbb{R}^m$.

The stabilized dual problem is constructed in such a way that the dual solution is restricted to a non-empty box strictly containing the known optimal dual solution. The stabilization method amounts to penalizing dual variables when they lie outside the predefined box, and enforces the selection of a valid optimal primal solution of the original problem (du Merle et al. (1999)). Computational experiments were run with instances, denoted as triplets (Beasley (1990)), because the optimal solution has exactly three items per bin, which fulfill exactly its capacity. If an instance of the CSP has no loss at optimality, the solution $\pi_i^* = \frac{w_i}{W}, i \in I$, is an optimal dual solution, because assigning these values to the dual constraints $\sum_{i \in I} a_{ip}\pi_i \leq 1$ simply replicates the knapsack constraint used to build the feasible patterns, and the corresponding dual objective function reaches the optimal value $\sum_{i=1}^{m} b_i w_i / W$ (Ben Amor (1997)). The computational results are impressive. The speed-up factor is approximately 10.0.

Computational results also show that convergence for different equivalent primal models is similar provided that their dual optimal spaces are equally restricted. Ben Amor (2002) compares two models for the CSP problem: in the first, the aggregated CSP, items of the same size were aggregated in the same constraint, as is usually done, while, in the second, the binary disaggregated CSP, items are considered in separate constraints, but dual inequalities impose equal dual values for items of the same size. The number of columns generated in both cases is remarkably similar, even though the models have very different sizes.

Finally a proximal stabilized column generation algorithm proved to be very efficient in solving CSP linear relaxation (Ben Amor (2002)). The key issue is that the dual vector $\pi^*$ defined above is a good initial solution for difficult problems (those ones with very small loss), and even it is not very close to a dual optimal solution for other problems, it may still have the nice property that the distribution of its components is close to the one of an optimal solution.

## 5 Extensions

One extension of the CSP is the multiple size lengths cutting stock problem (MLCSP) and its counterpart, the variable sized bin packing problems (VSBPP). They are variants of the standard problem in which large objects of different capacities are allowed. The arc flow model can also be extended to formulate these problems. Using a Dantzig-Wolfe decomposition, one obtains the machine balance problem formulation of Gilmore-Gomory (Valério de Carvalho (2002)). A branch-and-price algorithm for a version with limited availability of the bins, based on a master problem with columns of the reformulated model and branching constraints based on the arc-flow variables, proved to be adequate for both the MLCSP and the VSBPP (Alves and Valério de Carvalho (2003)).

Again, the advantage of aggregating items of the same size into a single group, as well as of the aggregation of bins, reducing symmetry, enables solving in a few seconds all the VSBPP instances proposed in (Monaci (2002)), in which a combinatorial enumeration algorithms failed to solve 65 out of the 300 instances within a time limit of 900 seconds. The larger instances have a maximum of 5 different types of bins and up to 500 items. Other classes of instances with about 25 different item sizes and 14 different bin capacities were also solved in, on average, one second, approximately. The algorithm has been applied to MLCSP instances, proposed in (Belov (2003)), which uses a forbidden columns dynamic programming enumeration scheme, providing comparable results; it was able to solve 44 out of 50 instances with 4 different types of bins with capacities ranging from 5000 to 10000.

Another extension of the one-dimensional CSP is a version in which the number of setups is minimized. It is a problem of great practical importance, because there are usually significant setup cost associated to changing from one cutting pattern to another. This problem is much more complex than the classical one-dimensional CSP, because additional binary variables are needed to indicate when a given cutting pattern is selected, and the resulting model has much larger duality gaps. This problem had only been previously tackled with heuristics. A branch-and-price-and-cut algorithm was applied to instances with up to 200 items. Computational results show that optimal solutions were obtained in 12 out of 16 instances, while in the remaining, solutions were found within one unit of optimality (Vanderbeck (2000a)).

## 6 Future research

The stabilization of the solution of the LP relaxation of the CSP produces impressive results. Instances with a much larger number of different item sizes can now be tackled, and their LP solutions and the corresponding bounds found in reasonable time. The application of similar ideas to the branch-and-price phase requires further investigation. The experience with the CSP also shows that it may be worthwhile to investigate and characterize the structure of the dual optimal space of other integer-programming and combinatorial-optimization problems.

Models with original variables provide additional insight, that can be used to derive more balanced and powerful branching rules, as the ones that result from hyperplane branching, and primal cuts expressed in terms of the original variables. Many integer programming and combinatorial optimization problems can be represented as pure network models, or network models with side constraints. Models with the columns of the reformulated model with branching constraints based on the variables of the original variables seem to be a promising approach for branch-and-price algorithm for this type of problems.

## References

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, Englewood Cliffs, New Jersey.

Alves, C. and Valério de Carvalho, J.M. (2003). A stabilized branch-and-price algorithm for integer variable sized bin-packing problems. Technical report, Dept. Produção e Sistemas, Universidade do Minho, Portugal, available at
http://www.dps.uminho.pt/cad-dps.

Barnhart, C., Johnson, E.L., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-Price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329.

Beasley, J.E. (1990). Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1060–1072.

Belov, G. (2003). *Problems, Models and Algorithms in One- and Two-Dimensional Cutting.* PhD thesis, Fakultät Mathematik und Naturwissenschaften der Technischen, Universität Dresden, Dresden, Germany.

Ben Amor, H. (1997). Résolution du problème de découpe par génération de colonnes. Master's thesis, École Polytechnique de Montréal, Canada.

Ben Amor, H. (2002). *Stabilisation de l'algorithme de génŕation de colonnes.* PhD thesis, École Polytechnique de Montréal, Canada.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J.M. (2003). Dual-optimal inequalities for stabilized column generation. Les Cahiers du GERAD G-2003-20, HEC Montréal, Canada.

Chan, L., Simchi-Levi, D., and Bramel, J. (1998). Worst case analysis, linear programming and the bin-packing problem. *Mathematical Programming*, 83:213–227.

Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8:101–111.

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

Degraeve, Z. and Schrage, L. (1999). Optimal integer solutions to industrial cutting stock problems. *INFORMS Journal on Computing*, 11:406–419.

Desrochers, M., Desrosiers, J., and Salomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354.

Desrosiers, J., Dumas, Y., Salomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In *Handbooks in Operations Research & Management Science 8, Network Routing*, pages 35–139. Elsevier Science B. V.

Dowsland, K.A. and Dowsland, W.B. (1992). Packing problems. *European Journal of Operational Research*, 56:2–14.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Math.*, 194:229–237.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159.

Dyckhoff, H. and Finke, U. (1992). *Cutting and Packing in Production and Distribution: a typology and bibliography.* Physica-Verlag, Heidelberg.

Dyckhoff, H., Scheithauer, G., and Terno, J. (1997). Cutting and packing. In *Annotated Bibliographies in Combinatorial Optimization*, pages 393–413. John Wiley and Sons, Chichester.

Farley, A. (1990). A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research*, 38:992–993.

Geoffrion, A. (1974). Lagrangian relaxation and its uses in integer programming. *Mathematical Programming Study*, 2:82–114.

Gilmore, P. (1979). Cutting stock, linear programming, knapsacking, dynamic programming and integer programming, some interconnections. In *Annals of Discrete Mathematics 4*, pages 217–236. North Holland, Amsterdam.

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

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

Goulimis, C. (1990). Optimal solutions to the cutting stock problem. *European Journal of Operational Research*, 44:197–208.

Helmberg, C. (1995). Cutting aluminium coils with high lengths variabilities. *Annals of Operations Research*, 57:175–189.

Kantorovich, L. (1960). Mathematical methods of organising and planning production (translated from a report in russian, dated 1939). *Management Science*, 6:366–422.

Kelley Jr., J.E. (1961). The cutting-plane method for solving convex programs. *J. Soc. Ind. Appl. Math.*, 8(4):703–712.

Marcotte, O. (1985). The cutting stock problem and integer rounding. *Mathematical Programming*, 33:82–92.

Marcotte, O. (1986). An instance of the cutting stock problem for which the rounding property does not hold. *Operations Research Letters*, 4:239–243.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, New York.

Monaci, M. (2002). *Algorithms for Packing and Scheduling Problems.* PhD thesis, Università Degli Studi di Bologna, Bologna, Italy.

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

Rietz, J. and Scheithauer, G. (2002). Tighter bounds for the gap and non-IRUP constructions in the one-dimensional cutting stock problem. *Optimization*, 51(6):927 – 963.

Scheithauer, G. and Terno, J. (1995). The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, 84:562–571.

Scholl, P., Klein, R., and Juergens, C. (1997). Bison: a fast hybrid procedure for exactly solving the one-dimensional bin-packing problem. *Computers and Operations Research*, 24:627–645.

Schwerin, P. and Wäscher, G. (1999). A new lower bound for the bin-packing problem and its integration into MTP. *Pesquisa Operacional, Special Issue on Cutting and Packing Problems*, 19:111–129.

Sessions, J., Layton R. and Guanda, L. (1988). Improving tree bucking decisions: A network approach. *The Compiler*, 6:5–9.

Sessions, J., Olsen E. and Garland, J. (1989). Tree bucking for optimal stand value with log allocation constraints. *Forest World*, 35:271–276.

Shapiro, J. (1968). Dynamic programming algorithms for the integer programming problem I: The integer programming problem viewed as a knapsack type problem. *Operations Research*, 16:103–121.

Simchi-Levi, D. (1994). New worst-case results for the bin-packing problem. *Naval Research Logistics*, 41:579–585.

Stadtler, H. (1990). One–dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research*, 44:209–223.

Sweeney, P. and Paternoster, E. (1992). Cutting and packing problems: a categorized, application-orientated research bibliography. *Journal Operational Research Society*, 43:691–706.

Valério de Carvalho, J.M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Ann. Oper. Res.*, 86:629–659.

Valério de Carvalho, J.M. (2002). LP models for bin-packing and cutting stock problems. *European J. Oper. Res.*, 141(2):253–273.

Valério de Carvalho, J.M. (2003). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing (in press).*

Valério de Carvalho, J.M. and Guimarães Rodrigues, A.J. (1995). An LP based approach to a two–phase cutting stock problem. *European Journal of Operational Research*, 84:580–589.

Vance, P. (1998). Branch-and-Price algorithms for the one-dimensional cutting stock problem. *Computational Optimization and Applications*, 9:211–228.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin-packing and cutting stock problems. *Mathematical Programmming, Ser. A*, 86:565–594.

Vanderbeck, F. (2000a). Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, 48:915–926.

Vanderbeck, F. (2000b). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48:111–128.

Wäscher, G. and Gau, T. (1996). Heuristics for the integer one-dimensional cutting stock problem: a computational study. *OR Spektrum*, 18:131–144.