Pergamon

Int. Trans. Opl Res. Vol. 5, No. 1, pp. 35–44, 1998

Printed in Great Britain
S0969-6016(97)00044-0
0969-6016/98 $19.00 + 0.00

Exact Solution of Cutting Stock Problems Using Column Generation and Branch-and-Bound*

J. M. VALÉRIO DE CARVALHO
Departamento Produção e Sistemas, Universidade do Minho, 4719 Braga, Portugal

This paper describes an attempt to solve the one-dimensional cutting stock problem exactly, using column generation and branch-and-bound. A new formulation is introduced for the one-dimensional cutting stock problem that uses general integer variables, not restricted to be binary. It is an arc flow formulation with side constraints, whose linear programming relaxation provides a strong lower bound. In this model, a cutting pattern, which corresponds to a path, is decomposed into single arc variables. The decomposition serves the purpose of showing that it is possible to combine the branch-and-bound method with variable generation. Computational times are reported for one-dimensional cutting stock instances with a number of orders up to 30. © 1998 IFORS. Published by Elsevier Science Ltd.

Key Words: cutting stock, column generation, branch-and-bound

INTRODUCTION

The one-dimensional cutting stock problem under consideration 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 rof b_i rolls of width w_i, $i = 1,2,\ldots,m$.

According to Dyckhoff's system (Dyckhoff, 1990), this problem is classified as 1/V/I/R, which means that it is a one-dimensional problem with an unlimited supply of rolls of identical size and a set of orders that must be fulfilled. The second entry in the classification, V, comes from *Verladeproblem*, which, in German, stands for problems where all items have to be combined to patterns which are assigned to a selection of large objects. In this case, the large objects are identical. The last entry means that there are relatively few different figures, even though the number of demanded items may be very large. This typically happens in cutting stock problems where there is a small set of different item sizes which are ordered in large quantities.

A combination of orders in the width of the roll is called a cutting pattern. Let x_j be a decision variable that designates the number of rolls to be cut according to cutting pattern j. The A matrix describes each cutting pattern, that is, each column $A_j=(a_{1j},\ldots,a_{ij},\ldots,a_{mj})^T$ defines a cutting pattern. The element a_{ij} represents the number of rolls of width w_i obtained in cutting pattern j.

The cutting stock problem is an integer programming problem that can be modelled as follows:

$$\min \sum_{j\in J} x_j \tag{1}$$

$$\text{Subj. to} \sum_{j\in J} a_{ij}x_j \geq b_i,\ i = 1,2,\ldots,m \tag{2}$$

$$x_j \geq 0 \text{ and integer},\ \forall j \in J \tag{3}$$

where J is the set of valid cutting patterns. For the cutting pattern to be valid:

*Paper presented at the 14th Triennial Conference of Operational Research Societies, Vancouver

$$\sum_{i=1}^{m} a_{ij} w_i \leq W$$

$$a_{ij} \geq 0 \text{ and integer, } \forall j \in J$$

The solution of the linear programming relaxation of this model may present some difficulties. If, for small instances, it is possible to enumerate all the possible patterns and to find the optimal solution, for larger instances, this method is impractical. Even for moderately sized instances, the number of columns can be in the order of the hundreds of million. To tackle this problem, Gilmore and Gomory (1963) introduced a column generation procedure. At each iteration, the information given by the dual variables is used to evaluate which columns are still attractive to the main problem (restricted master problem). Usually, the single most attractive column is added to the main problem. It can be determined by solving a subproblem, which is a knapsack problem.

To obtain integer solutions to the cutting stock problem, heuristic methods are usually used. Just to mention a few examples, Gilmore and Gomory (1963) suggested rounding up and down the fractional solution as a fast method of obtaining a discrete solution. Stadtler (1990) used a more elaborate procedure in a problem arising in the aluminium industry. The heuristic, at each iteration, rounds up the variable with the larger fractional part, fixes that variable and reoptimizes the problem. If overproduction occurs, some variables are decremented, which can lead to underproduction. In this case, the missing orders are combined using a first fit decreasing (FFD) heuristic.

There are also some references with exact methods to approach the integer problem. In a problem where the ratio between the width of the rolls and the average width of the orders is small, being the number of possible cutting patterns, typically, a couple of hundreds, Goulimis (1990) enumerated the entire set of columns and used the Chvátal – Gomory cutting plane technique, resorting to branch-and-bound when necessary. In a different setting, Vance *et al.* (1994) developed an exact procedure, based on column generation and branch-and-bound, for the binary cutting stock problem. In the binary case, the demand for each order is restricted to be equal to one.

Procedures that combine column generation and branch-and-bound were used in other problems, as, for instance, in a routing problem with time windows (Desrosiers *et al.*, 1984) and in the edge colouring problem (Nemhauser and Park, 1991). In general, the implementation of procedures that combine the two methods has to overcome a crucial difficulty: the problem looses its "structure" as the branching constraints are added to the restricted master problem; and the type of subproblem that has to be solved in each iteration may change.

In the case of the cutting stock problem, for the formulation under consideration, after solving the root node of the search tree, which corresponds to solving the linear programming relaxation, it may be necessary to introduce branching constraints in the restricted master problem. The column generated while solving the linear programming relaxation may be sufficient to obtain an integer solution with an objective value equal to the integer round up of the objective value of the fractional solution, which clearly guarantees that the integer solution is optimal. However, it may happen that, deep in the branch-and-bound tree, a set of columns that were not generated is necessary to obtain the optimal solution for that node, and if that set is missing the solution obtained may be non-optimal. Therefore, it may be necessary to generate new columns during the branch-and-bound phase.

The generation of columns at each node of the branch-and-bound tree occurs after the introduction of branching constraints in the restricted master problem. Under these circumstances, the subproblem is changed, and the optimal solution of the original knapsack problem may be a column that should not be introduced in the restricted master problem. In fact, it may happen that a particular column that was set to zero by a branching contraints turns out to be most attractive column generated by the subproblem. To overcome this difficulty, Vance *et al.* (1994) added generalized upper bounding constraints to the knapsack subproblem, and solved the modified problem, but eventually the subproblem completely looses its structure and has to be solved as a general integer programming problem.

In this article, an arc flow formulation with side constraints for the cutting stock problem is introduced. The model has a set of flow conservation constraints and a set of constraints to ensure that the demand is satisfied. The corresponding path flow formulation is equivalent to the classical formulation for the cutting stock problem. Path formulations are usually preferred to arc formulations, because they require less computational space [see Ahuja *et al.* (1993) chapter 17].

Arc flow formulation was used because it allows column generation at any node in the branch-and-bound tree, and the implementation of the algorithm involves modifications to the subproblem, after solving the linear programming model, that are conceptually simpler. During the solution of the linear programming relaxation, to accelerate the column generation procedure, instead of producing the most attractive arc, the subproblem generates sets of arcs, which correspond to valid paths, viz. cutting patterns.

Arc flow formulations usually have a large number of flow conservation constraints. A key issue is that the sets of arcs can be evaluated without considering explicitly these constraints. Actually, the model starts with all the flow conservation constraints relaxed, and only those that correspond to the arcs introduced in the restricted master problem are considered. A significant number of flow conservation constraints has never to be considered during the solution.

In Section 2, the formulation is introduced and some criteria are presented to reduce the number of variables and also the valid inequalities that are used to tighthen the formulation. Section 3 describes the solution of the linear programming relaxation using a column generation procedure, and shows the subproblem to be solved. In Section 4 the branch-and-bound procedure, the branching rules and the articulation of this process with the column generation procedure are presented. Section 5 describes some details of the implementation and the computational results obtained in the solution of some test problems.

MATHEMATICAL FORMULATION

Given rolls of an integer width W and a set of orders of width $w_1, w_2, \ldots, w_m$, the problem of determining a single valid cutting pattern can be modelled as the problem of finding a path in an acyclic directed graph with $W + 1$ vertices. Consider a graph $G = (V, A)$ with $V = \{0, 1, 2, \ldots, W\}$ and $A = \{(k, l)\colon 0 \leq k < l \leq W$ and $l - k = w_i$ for every $i \leq m\}$, meaning that there exists a directed arc between two vertices if there is an order of the corresponding width.

Consider additional arcs between $(k, k + 1)$, $k = 0, 1, \ldots, W - 1$ corresponding to unoccupied portions of the roll, that is, trim loss. There is a valid cutting pattern iff there is a path between vertices 0 and W. The arcs that constitute the path define the cutting pattern.

Example 2.1. Figure 1 shows the graph associated with an instance with $W = 5$ and orders of width 3 and 2. In the same figure, a path is shown that corresponds to 2 units of width 2 and 1 unit of trim loss.□

This kind of formulation has already been used to model knapsack problems as the problem of determining the longest path in a directed graph (Papadimitriou and Steiglitz, 1982). Likewise, it can be used to model cutting stock problems. If one cutting pattern corresponds to the flow of one unit between vertices 0 and W, a path carrying a larger flow will correspond to cutting the same cutting pattern multiple times.

By the flow decomposition properties [see, for instance Ahuja *et al.* (1993)], non-negative flows can be represented by paths and cycles. The graph G is acyclic and, therefore, any flow can be decomposed in directed paths connecting the only excess node (node 0) to the only deficit node (node W).

The problem is formulated as the problem of determining the minimum flow between vertice 0 and vertice W with additional constraints enforcing that the sum of the flows in the arcs of each order must be greater or equal to the demand. Consider decision variables x_{ij}, associated with the arcs defined above, which correspond to the number of items of width $j - i$ placed in any cutting pattern at the distance of i units from the left border of the roll.

$$\min z \tag{4}$$

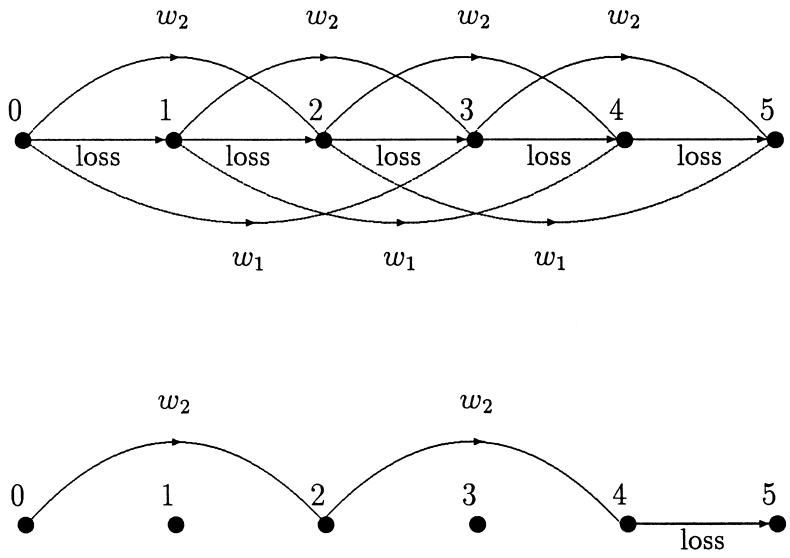

Fig. 1. Graph and a cutting pattern.

$$\text{subj. to} + \sum_{i|(i,\,j)\in A} x_{ij} - \sum_{k|(\,j,\,k)\in A} x_{jk} = \begin{cases} -z, & \text{if } j = 0 \\ 0, & \text{if } j = 1, 2, \ldots, W-1 \\ z, & \text{if } j = W \end{cases} \tag{5}$$

$$\sum_{k|(k,\,k+w_i)\in A} x_{k,\,k+w_i} \geq b_i,\ i = 1, 2, \ldots, m \tag{6}$$

$$x_{ij} \geq 0 \text{ and integer}, \forall (i,j) \in A \tag{7}$$

The matrix that defines this problem does not have, in general, any special property that guarantees that the basic solutions obtained are integer, and it may be necessary to resort to branch-and-bound after solving the linear programing relaxation. The solution thus obtained has integer values of flow in every arc, and, by the flow decomposition property referred above, can be transformed into an integer solution to the one-dimensional cutting stock problem.

The lower bound provided by the linear programming relaxation of this model is as good as the one provided by the classical formulation:

Preposition 2.1. The optimum of the linear programming equations [4]–[7] is at least as good as the optimum of the linear relaxation of the classical model in equations [1]–[3].

Proof. It follows from the fact that the optimal fractional solution to equations [4]–[7] gives a valid solution to formulation equations [1]–[3] which, being a minimization problem, will have an optimum equal or smaller.□

This bound is known to be very tight. Most of the cutting stock instances have gaps smaller than one, which is commonly referred as the integer round-up property (Marcotte, 1985, 1986). Nevertheless, several instances and families of instances, are known with gaps slightly >1 (Fieldhouse, 1990). Using the criteria referred below, that strengthen the formulation, the duality gap was bridged and gaps >1 have never arisen.

Enumeration of columns and reduction criteria

Using the variables presented thus far, for each cutting pattern, there are many alternative solutions with exactly the same items, but placed in different positions. Applying the Criteria presented below, the solution in which the items are placed in the cutting pattern in positions that correspond to decreasing values of width is searched, thus reducing the symmetry of the solution space and the size of the model.

Criterion 1. An arc of order j, designated by $x_{k,\,k\,+\,wj}$, can only have its tail at a node k that is the head of another arc of order i, $x_{k\,-\,i,\,k}$, for $w_i > w_j$, or else, from node 0, that is, the left border of the roll.

In particular, if the cutting pattern has any trim loss, because items are to be placed in order of decreasing width, the trim loss will appear last in the cutting pattern. A cutting pattern can never start with trim loss.

Criterion 2. All the trim loss arcs $x_{i,\,i\,+\,1}$ can be set at zero for $i < w_m$.

Criterion 3. In a cutting pattern, the number of consecutive arcs corresponding to a single order must be smaller or equal to the required demand for that order.

Let z_{LP} be the optimal value of the linear programming relaxation and $\lceil z_{\mathrm{LP}} \rceil$ the smallest integer greater or equal to z_{LP}. After solving the linear programming relaxation, the model can be strengthened by introducing a valid inequality that forces the number of trim loss arcs to be greater than or equal to $T_{\min}$.

Definition 2.1. The minimum trim loss $T_{\min} = \lceil z_{\mathrm{LP}} \rceil W - \sum_{i=1}^{m} w_i b_i$.

The loss will be equal to this value if the instance has the integer round-up property, as it happens in the generality of the cutting stock instances.

Proposition 2.2. The constraint $\sum_{(k,\,k+1)\in A} x_{k,\,k+1} \geq T_{\min}$ is a valid inequality for the integer programming problem.

Example 2.2. Consider a cutting stock problem with rolls of width $W = 7$ and demands $w = (5, 3, 2)$, and quantities $b = (1, 3, 2)$. The linear programming formulation of this instance and the corresponding underlying graph are shown in Fig. 2. The model has a redundant constraint concerning trim loss that can be strengthened after solving the linear programming relaxation.

According to Criterion 3, it should be pointed out that the arc x_{46} is not a valid arc for this particular instance, because the demand for items of width 2 is only 2. These items must be placed after rolls of width 5 or 3, or at the beginning of the cutting pattern, from vertex 0. In either case, an item of width 2 can never be placed at a distance of 4 units from the left border of the roll. The number of flow conservation constraints can be smaller than the value of the width of the rolls. In this example, there is no flow conservation constraint for vertex 1.

The solution of the linear programming relaxation has an optimal value of 2.75. This instance has the integer round-up property, and its optimal solution is equal to 3. Therefore, the minimum trim loss $T_{\min} = 3$. If an optimal solution with the integer round-up property is searched, then the variables $x_{23} = x_{34} = 0$ can be fixed, and the trim loss inequality can be tightened, lifting its right hand side, which becomes $x_{45} + x_{56} + x_{67} \geq 3$.□

It can be shown that there exists a solution to the linear programming relaxation in which all the demand constraints are observed without slack (de Carvalho, 1995). As a corollary, this valid inequality, which imposes a lower bound to the value of the trim loss, forces the objective value to be integer.

As a rule, it is not possible to establish an upper bound on the gap between the values of the optimal solutions of the integer problem and its linear programming relaxation. In the one-dimensional cutting stock problem, it happens that almost all the instances have a gap smaller than unity. This property is known as the integer round-up property (Marcotte, 1985). Therefore, most of the instances will have a trim loss equal to $T_{\min}$. However, if the instance does not have the integer round-up property, the amount of loss has to be increased by W units.

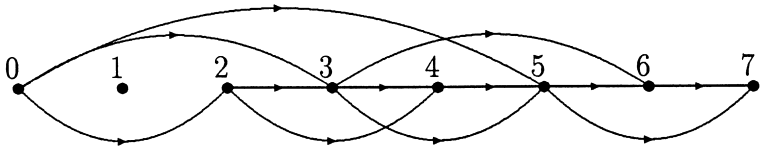

Fig. 2. Linear programming model and the underlying graph.

LINEAR PROGRAMMING RELAXATION

At each iteration, a subproblem is solved and a set of arcs (columns) is introduced in the restricted master problem. Let u_j, $j = 0, 1, \ldots, W$, be the dual variables associated with the flow conservation constraints and v_k, $k = 1, 2, \ldots, m$, the dual variables associated with the demand constraints. Each column, which corresponds to arc (i, j), has only three nonzero elements, that is, $a-1$ in row i, $a+1$ in row j and $a+1$ in the demand row k, corresponding to the ordered width $w_k = j - i$.

Let B be the current basis of the restricted master problem and $c_B B^{-1}$ the corresponding dual solution. Let A_{ij} be the column corresponding to variable x_{ij}. Column A_{ij} is attractive if its reduced cost $c_B B^{-1} A_{ij} - c_{ij}$ is > 0, being all $c_{ij} = 0$. The reduced cost of a single column is equal to $\bar{c}_{ij} = -u_i + u_j + v_k$.

To accelerate the generation procedure, instead of generating a single arc, sets of arcs are generated at each iteration. Let $P = (0, a)(a, b)\ldots(y, z)(z, W)$ be a directed path starting at node 0 and ending at node W.

Proposition 3.1. The reduced cost of a path P is equal to $\bar{c}_P = \sum_{(i,j)\in P} v_k - 1$.

Proof. The reduced cost of a path $\bar{c}_P = \sum_{(i,j)\in P} \bar{c}_{ij} = \sum_{(i,j)\in P} -u_i + u_j + v_k$. The flow conservation dual variables cancel out for all nodes, except for the two terminal nodes. Therefore, $\bar{c}_P = -u_0 + u_W + \sum_{(i,j)\in P} v_k$. The two former terms correspond to a column which is symmetrical to column z, and, therefore, contribute to the reduced cost with a value of -1.□

Corollary 3.1. The set of columns that correspond to path P is attractive if $\sum_{(i,j)\in P} v_k > 1$.

To determine the most attractive set of columns, a subproblem is solved, which is the longest path in an acyclic digraph with arc costs that depend on the value of the dual variables. This result has a well known equivalent in the classical cutting stock formulation. It is important because it allows the evaluation of a set of columns without considering explicitly the flow conservation constraints.

At each iteration, if an arc (i, j) is part of an attractive path, the algorithm checks if the flow conservation constraints for nodes i and j, respectively, have already been considered. If not, new constraints are considered in the model. The model grows in two directions as columns and flow conservation constraints are added to it.

BRANCH-AND-BOUND

The procedure developed is oriented to a problem where the gaps are almost always strictly < 1, and can be reduced to zero by introducing the valid inequality referred in Proposition 2.2. The optimization problem is solved as a sequence of decision problems, in which we want to find if there is, or there is not, an integer solution with an objective value equal to the smaller known lower bound, which will be always an integer value, denoted as Z_{LB}.

The first decision problem to be solved is to determine if there is an integer solution of value $\lceil z_{\mathrm{LP}} \rceil$, that is, $Z_{\mathrm{LB}} = \lceil z_{\mathrm{LP}} \rceil$. As we know, most of the cutting stock problem instances have the integer round-up property, meaning that almost always the solution to this decision problem will provide an optimal integer solution to the optimization problem.

However, if an integer solution is not found in the search tree, meaning that the instance does not have the integer round-up property, the lower bound has to be increased by one unit, that is, $Z_{\mathrm{LB}} = Z_{\mathrm{LB}} + 1$, and the procedure has to be repeated. Clearly, the first integer solution found is optimal. The application of this method can only be justified in problems with minuscule gaps, and its use is not reasonable in general integer programming problems.

Branching constraints are imposed on single variables of the master problem, starting from variables that correspond to larger widths and are placed nearer the left border of the roll, that is, the fractional variable with the smaller value i, and to break ties the one with larger $\{j - i\}$. A depth first search is performed, using branching constraints of the following type:

$$x_{ij} \geq [x_{ij}] \text{ and } x_{ij} \leq [x_{ij}]$$

In each node of the search tree, there is a need to determine if there is any solution with objective value Z_{LB}. If not, the node is fathomed. This can only be done when the generation procedure fails to produce more attractive variables and the objective function value of the restricted master problems is strictly larger than Z_{LB}.

After a branching constraint is added, the restricted master problem is reoptimized, and one of the following cases occurs:

1. the solution is integer, with value equal to Z_{LB}, which means that it is optimal;
2. the solution is fractional, with value equal to Z_{LB}, being necessary to introduce new branching constraints;
3. the solution has a value that is strictly greater than Z_{LB}. The column generation procedure is called trying to reach a solution with value Z_{LB}, leading either to case (1) or (2). If this is not possible, the node is fathomed.

Figure 3 presents a flowchart for the column generation/branch-and-bound procedure.

During the branch-and-bound phase, there may be arc variables that have to be brought to the restricted master problem to guarantee that an optimal solution is found. The branching constraints are imposed on the variables that belong to the restricted master problem. Therefore, the expression for the evaluation of the reduced cost of the variables that are out of the restricted master problem is not changed. At any node of the branch-and-bound tree, under case (3) described above, arcs (variables) may be priced out, and those that are attractive are introduced in the restricted master problem.

Depending on the set of branching constraints introduced in the restricted master problem, some variables can be ignored according to the following criteria.

Criterion 1. Variables (arcs) set to zero in the restricted master problem should not be regenerated by the subproblem.

Criterion 2. Suppose there is a lower bound on a single arc, or a set of lower bounds on arcs that belong to the same order, which sum up the required demand. As a solution is searched without overproduction, all the remaining arcs belonging to that order can be set to zero in the restricted master problem, and not regenerated by the subproblem.

In the cutting stock problem, a cutting pattern corresponds to a path between vertices 0 and W. The decomposition of the path into single arcs serves the purpose of showing that it is possible to combine the branch-and-bound method with variable generation to solve cutting stock problems exactly. It can be used to solve problems with general integer variables, not restricted to being binary.

COMPUTATIONAL RESULTS

The algorithm was tested in a set of test problems provided by Wäscher and Gau (1993). The instances selected have orders of integer width. Although the model can be used in instances with rational values, the resulting models are larger, because it is necesssary to discretize the values. The solution of some instances are reported from the PIE group, which have integer widths up to 200 and a number of orders up to 30. The number of orders and the width of the rolls are indicated in the first two columns in Table 1.

The procedure calls the XMP routines, described in Marsten (1981) which use the revised simplex method and the factorization of the inverse. Lower and upper bounds are dealt implicitly, which enables an easy implementation of the branch-and-bound procedure.

Computational tests were run in a 40 MHz PC 486 DX, with 4 Mb of RAM memory. The algorithms were coded in Fortran and compiled using Salford FTN77 under DBOS (Salford, 1995).

During a pre-processing phase, all the possible arcs are enumerated according to the criteria presented in Section 2. Then, the initial solution is set. For each order, the following variables are selected: $x_{0k}, x_{k,\,2k}, \ldots, x_{(s-1)k,\,sk}$, where $sk \leq W$ and $(s+1)k > W$. For each instance, the

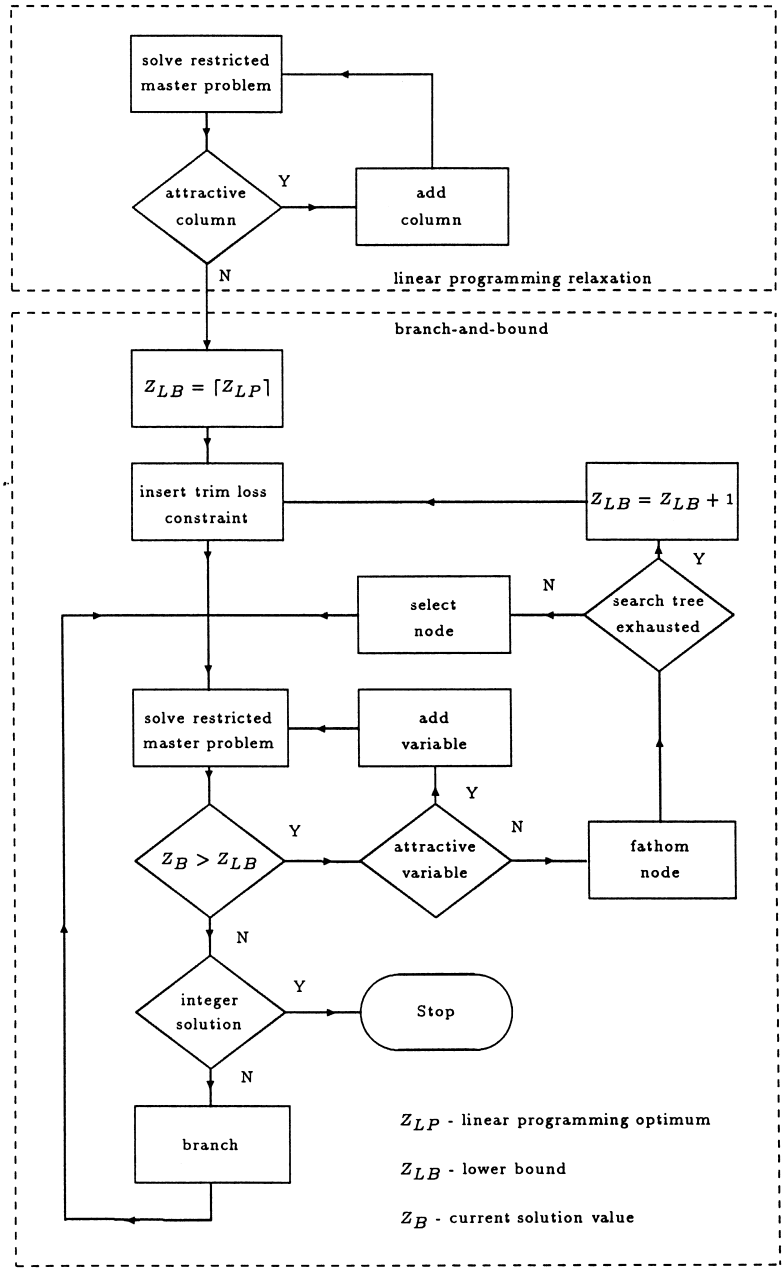

Fig. 3. Column generation/branch-and-bound procedure.

Table 1. Characteristics of the problems

	Orders	Width	Arcs	Constraints
PIE64-1	7	215	501	209
PIE64-2	7	215	501	209
PIE64-3	7	215	707	219
PIE64-4	7	215	707	219
PIE64-5	10	148	376	131
PIE64-6	20	150	268	124
PIE64-7	20	150	639	144
PIE64-8	30	182	250	146
PIE64-9	30	182	250	146
PIE64-10	30	182	1226	186
PIE64-11	30	182	1226	186

Table 2. Solution data

	cols lp	cols bb	vért bb	t_{pp}	t_{lp}	t_{bb}	t_{tot}	z^*
PIE64-1	73	0	1	0.11	7.75	0.49	8.35	640
PIE64-2	90	0	4	0.11	10.11	0.88	11.10	245
PIE64-3	131	0	11	0.16	17.25	2.97	20.38	493
PIE64-4	46	0	9	0.17	7.86	1.98	10.00	197
PIE64-5	52	0	4	0.16	3.35	0.44	3.96	239
PIE64-6	40	0	0	0.11	2.31	0.00	2.42	1527
PIE64-7	65	0	43	0.11	5.28	4.18	9.56	1000
PIE64-8	58	42	2	0.11	3.35	3.46	6.92	2845
PIE64-9	58	42	2	0.11	3.35	3.46	6.92	2575
PIE64-10	127	0	45	0.17	14.12	9.73	24.01	1390
PIE64-11	164	0	49	0.16	18.46	10.44	29.06	1242
Average	82.2	7.6	15.5	0.13	8.47	3.46	12.06	

number of arcs (variables) and the number of constraints, which are determined by enumeration, are indicated in the last two columns in Table 1.

In Table 2, the data related to the solution is presented. It can be verified that only a small fraction of the possible columns are used in the solution. The meaning of each column is as follows:

cols lp	Number of columns generated while solving the linear programming relaxation;
cols bb	Number of columns generated during the branch-and-bound phase;
vert bb	Number of vertices explored in the branch-and-bound phase;
t_{pp}	Preprocessing time (seconds);
t_{lp}	Linear relaxation solution time (seconds);
t_{bb}	Branch-and-bound time (seconds);
t_{tot}	Total time (seconds);
z^*	Optimum value.

In most instances, it was not necessary to generate extra columns during the branch-and-bound phase, because the columns generated while solving the linear programming relaxation were sufficient to build an integer optimal solution.

CONCLUSIONS

A new model and an exact procedure for the one-dimensional cutting stock problem was presented. Its application is not restricted to the binary case, where all the quantities demanded are equal to one.

As the relaxation is very tight, the branch-and-bound tree are tractable. The formulation is particularly sensitive to the width of the rolls, because the size of the subproblem and of the restricted master problem grow with the width of the rolls.

The dimensions of the restricted master problem grow in two directions. At first, all the flow conservation constraints are relaxed. As the algorithm proceeds, sets of variables are added to the main problem and the flow conservation constraints that correspond to those variables are considered.

Acknowledgements—This work was partially supported by the Research Center Algoritmi of the University of Minho, and was developed in the Industrial and Systems Engineering Group.

REFERENCES

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

de Carvalho, J. M. V. (1995) A model for the one-dimensional cutting stock problem. Unpublished manuscript, 1995.

Desrosiers, J., Soumis, F. and Desrochers, M. (1984) Routing with time windows by column generation. *Networks* **14,** 545–565.

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

Fieldhouse, M. (1990) The duality gap in trim problems. *SICUP Bulletin* 5, 4–5.
Gilmore, P. C. and Gomory, R. E. (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.
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.
Marsten, R. M. (1981) The design of the XMP linear programming library. *ACM Transactions on Mathematical Software* **7,** 481–497.
Nemhauser, G. and Park, S. (1991) A polyhedral approach to edge coloring. *Operations Research Letters* **10,** 315–322.
Papadimitriou, C. H. and Steiglitz, K. (1982) *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ.
Salford (1995) FTN77 Salford Software Ltd. Adelphi House, Adelphi Road, Salford M3 6EN, U.K.
Stadtler, H. (1990) One-dimensional cutting stock problem in the aluminium industry and its solution. *European Journal of Operational Research* **44,** 209–224.
Vance, P., Barnhart, C., Johnson, E. L. and Nemhauser, G. L. (1994) Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications* 111–130.
Wäscher, G. and Gau, T. (1993) Test data for the one-dimensional cutting stock problem. Arbeitsbericht No. 93/9, October.